Probabilistic Planning for Robotics with ROSPlan*

Gerard Canal[1], Michael Cashmore[2], Senka Krivić[2], Guillem Alenyà[1], Daniele Magazzeni[2], and Carme Torras[1]

[1] Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
{gcanal,galenya,torras}@iri.upc.edu
[2] Department of Computer Science, King's College London, United Kingdom
name.surname@kcl.ac.uk

Abstract. Probabilistic planning is very useful for handling uncertainty in planning tasks to be carried out by robots. ROSPlan is a framework for task planning in the Robot Operating System (ROS), but until now it has not been possible to use probabilistic planners within the framework. This systems paper presents a standardized integration of probabilistic planners into ROSPlan that allows for reasoning with non-deterministic effects and is agnostic to the probabilistic planner used. We instantiate the framework in a system for the case of a mobile robot performing tasks indoors, where probabilistic plans are generated and executed by the PROST planner. We evaluate the effectiveness of the proposed approach in a real-world robotic scenario.

1 Introduction

Planning for robotics means planning in dynamic and uncertain domains, in which the outcomes of actions can have a reasonable chance of failure, or non-deterministic effects, for example in complex manipulation tasks and navigation in crowded spaces. Probabilistic planning is an approach to plan under uncertainty, commonly meaning planning with probabilistic action effects. A probabilistic planner tries to maximize the probability of success of a plan.

In many domains it is possible to ignore the probabilistic nature of the environment by generating deterministic plans, and replanning when they fail during execution. However, for some problems it is advantageous to consider the probabilities: for example when there is more than one path to the goal and those paths have different associated rewards and probabilities of success, or the state-space includes dead-end states. Given different paths to a goal, the paths with higher associated rewards might counterintuitively be those that are longer, or otherwise the cost structure might be far from obvious. These kinds of problems are termed *probabilistically interesting* [20].

* This work has been supported by the ERC project Clothilde (ERC-2016-ADG-741930), the HuMoUR project (Spanish Ministry of Science and Innovation TIN2017-90086-R) and by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656). G. Canal is also supported by the Spanish Ministry of Education, Culture and Sport by the FPU15/00504 doctoral grant and the mobility grant EST17/00371. The research from KCL was partly supported by Korea Evaluation Institute of Industrial Technology (KEIT) funded by the Ministry of Trade, Industry & Energy (MOTIE) (No. 1415158956).

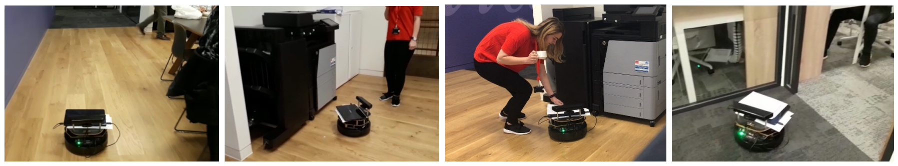


Fig. 1. The scenario in which we test the proposed system is an office environment. A mobile robot, the TurtleBot 2[4] is used for the print-fetching service. When the robot gets a request for fetching prints, it decides from which printer to collect them. Since it is not equipped with an arm, it asks a random nearby person to put prints on it, and delivers them to the user.

Robotics domains are often probabilistically interesting. For example, an autonomous robot in a dynamic environment can easily move into a state from which it does not have the capability to recover by itself, requiring human intervention. Therefore, robots are often expected to follow the slower, safer paths to the goal to avoid failure. However, by reasoning over the probabilities during planning, more efficient solutions can be found.

The Relational Dynamic Influence Diagram Language (RDDL) is a stochastic domain description language for probabilistic planning. The International Probabilistic Planning Competition (IPPC) uses RDDL [25] for probabilistic planning problems. RDDL is well-suited to describing probabilistically interesting problems, using a dynamic Bayes net formalism [8], as opposed to the effects-based (P)PDDL. Subsequently, both the first and second-place entries in the 2012 IPPC were planners that actively reasoned with probabilities: PROST [16], used in our experiments; and Glutton [17].

The ROSPlan framework [6] provides a standard approach for planning in the Robot Operating System (ROS). Until now, one drawback of ROSPlan is that it has been limited to deterministic and contingent planning, using PDDL2.1 [9], and is not suitable for probabilistic planning. The main contributions of this paper are: *(i)* A standardized integration of RDDL and ROSPlan, enabling the straightforward application of the probabilistic planning in robotic domains using ROS. *(ii)* A demonstration of a mobile robot autonomously generating and executing probabilistic plans using this integration in an extensible RDDL domain. We extend ROSPlan to handle RDDL semantics, produce RDDL problem instances, and interface with any RDDL planner that can be used with the RDDLSim server used in the IPPC. In addition, we extend the action interface of ROSPlan, which handles the execution of discrete actions, to reason with non-deterministic effects. To enable distinction between deterministic and non-deterministic effects, we identify two kinds of propositions: *sensed*, whose truth value can be sensed by the agent during execution, and so can be included within a probabilistic effect; and *non-sensed*, which can only produce deterministic effects.[5]

We test the system in a mobile robot scenario and define a challenging *print-fetching* domain where the robot is used as a service robot for fetching printed documents in an office (Figure 1). Human-robot interaction supplements the lack of manipulation

[4] `https://www.turtlebot.com/`

[5] The source code of the elements described in this paper can be found in the main ROSPlan repository `https://github.com/KCL-Planning/ROSPlan`

abilities of the used robot, thus allowing it to perform this task. A real-world evaluation is carried out in an environment with high uncertainty.

2 Related work

There are numerous approaches addressing uncertainty in the planning and execution process e.g. conformant planning [26], contingent planning [2] or replanning [29]. Other approaches use machine-learning techniques to decrease uncertainty in the planning problem, e.g. [7] learn probabilistic action models and [18] remove uncertainty in state prior to planning by making predictions based on initially known data. Also, there is work on building architectures that involve reactive components to cope with uncertainties in robotics domains [15]. ROSPlan has been used to perform planning for control of multiple-robot systems running with ROS [4, 6]. However, all of these works focus on purely deterministic planning.

Furthermore, probabilistic planning is a standard approach for planning with uncertainty in robotics. An overview of approaches to probabilistic planning is provided in [10]. The most common approach to planning with uncertainties in robotics is modelling the task as a Markov Decision Problem (MDP), optionally a partially-observable MDP (POMDP). In contrast to deterministic planning, notably the PDDL2.1 [9] formalism used so far with ROSPlan [6], robotics scenarios must often cope with exogenous and uncontrollable events [22], which can be easily modelled as POMDPs [21]. Solutions to the MDPs for robotics can form policies with finite horizon [3], adopt a satisficing approach [19], or maximize the probability of reaching a goal state [23]. RDDL [25] is well-suited for modelling problems of this kind. It is a dynamic Bayes net formalism, allowing for unrestricted concurrency. This is an essential component in robotics applications, in which the agent must execute the plan in a physical environment. For example, in multi-robot navigation scenarios in which motion is stochastic from the perspective of the planning model.

Atrash and Koenig [1] note that POMDP planning policy graph solutions are similar to the finite-state machines normally used for control. As a result, it has been applied successfully in many robotic use cases featuring uncertainty, such as robotic exploration missions [27]; or those with action outcomes that are inherently non-deterministic, such as manipulation problems [14], human-robot interaction [12] and physically assistive robotics [5]. The office setting is a common environment for autonomous service robots, and can exhibit these kinds of uncertainty. Examples are collaborative robots servicing human indoor environments [28] and an office-guide robot for social studies [24].

3 System Description

In order to include the ability of planning with probabilistic domains within the ROSPlan framework, we have designed and implemented a new Knowledge Base and problem generator that are able to handle probabilistic planning problems written in RDDL.

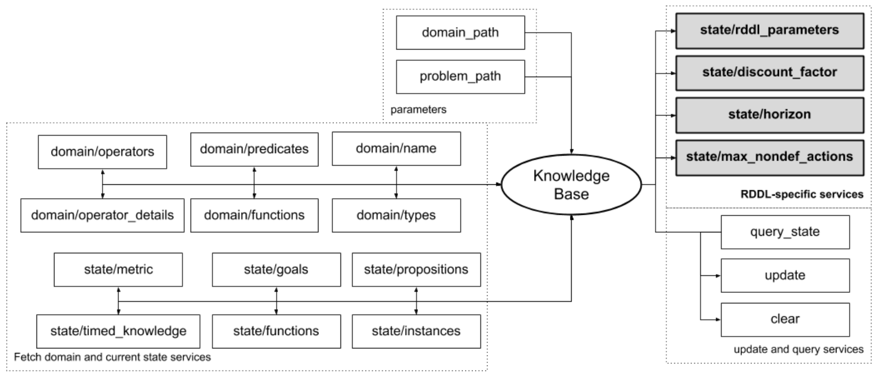


Fig. 2. ROSPlan's Knowledge Base interface. The RDDL services are highlighted.

RDDL Knowledge Base The Knowledge Base (KB) in ROSPlan stores the current model of the environment. It is an interface for updating and fetching a PDDL model in ROS, and primarily consists of a set of ROS services forming this interface. These services are used by many other components of ROSPlan, most of which require state or domain information, such as problem generation and plan execution and validation.

The integration of RDDL with the ROSPlan KB adheres to the existing interface for two reasons: to preserve compatibility with systems already using ROSPlan, and to allow for the interchange of RDDL and PDDL KBs. Therefore, the RDDL KB translates the RDDL domain and problems to PDDL-like structures. Given that RDDL is more expressive than PDDL, the RDDL KB also extends the interface with new ROS services providing RDDL-specific functionality. Figure 2 shows the extended KB interface.

To process the RDDL domain into a PDDL-like structure, action-fluents are mapped to PDDL operators, and state-action constraints (also called action-preconditions in newer versions) are encoded as PDDL preconditions in the following way:

1. The constraints are searched to find those of the form $action\text{-}fluent \rightarrow (formula)$.
2. When found, the right hand side is encoded as an action precondition.

We assume the $formula$ only includes conjunctions of state fluents. This is due to a current limitation of ROSPlan, which does not support quantified or disjunctive conditions in PDDL.

Action effects are obtained from conditional probability functions (*cpfs*). This block describes how each fluent changes at each time step, determined by the current state and actions. In order to obtain the effects of an operator, the *cpfs* block is processed for each action fluent. As a new feature, probabilistic effects are also considered and added to the Knowledge Base. We only consider probabilistic effects to be of the RDDL's Bernoulli distribution and Discrete distribution types. Stochastic effects are processed in a similar way to non-probabilistic ones, but when the result of the *cpf* expression is probabilistic, the effect is added to a new effect list with an associated probability formula. In order to provide information on exogenous effects, a new operator named *exogenous* is created. This operator has as its effects all the exogenous effects that may happen but are not related to any specific action-fluent. Effects of this kind are otherwise considered in the same way as the effects of other operators. Finally, the reward function is fully instantiated and represented as a PDDL metric function, with the metric set to be maximized. In the case where there is a state-fluent named "goal", its expression from the *cpfs* block will be included as the PDDL goal.

Although some assumptions are made, such as the conjunctive-only preconditions, it should be noted that these assumptions apply only to the RDDL domain file, which will not be modified when loaded into the KB. Instead, it is passed entirely to the planner. Therefore, although some elements of the domain may be unknown by the KB, the problem is entirely captured, and the planner will still provide correct plans.

Problem Generation The ROSPlan Problem Interface node is used to generate a problem instance. It fetches the domain details and current state through service calls to a Knowledge Base node and publishes a PDDL problem instance as a string, or writes it to file. To be able to use a planner with a RDDL input, a RDDL Problem interface has been implemented.

The generation of the RDDL problem requires checking operator effects to find which predicates change due to some operators (the state fluents) and which are static for the planning problem (called non-fluents). Additionally, the planning horizon and the discount factor are set by default, or from RDDL-specific services in the KB. A feature of this approach is that as the KB interface is common for both RDDL and PDDL, ROSPlan can generate problems independently of the which KB is used. Thus, a RDDL instance file can be generated from a PDDL instance and vice versa. The requirement is that that both domains share the same structure (i.e., operators and predicates). Therefore, it is now very simple to have both deterministic and probabilistic planners running together, for example, for plan checking and validation or in a planning system composed of both stochastic and deterministic planners.

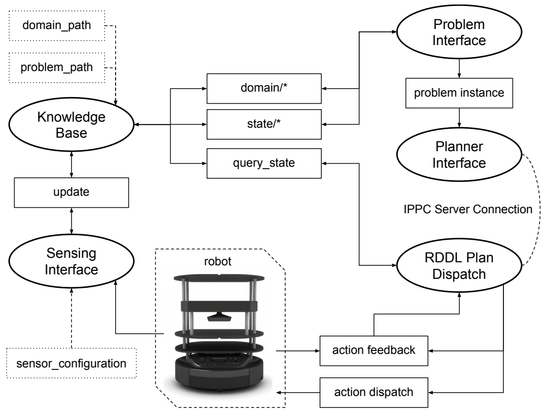


Fig. 3. The system architecture used in our scenario. ROS nodes are represented by ovals, and implement the ROSPlan interfaces. Message and service topics are represented by solid boxes, parameters by dotted boxes.

4 Online Planning and Execution with RDDL Planners

ROSPlan provides two plan dispatchers: the simple plan dispatcher for non-temporal, sequential plans, and the Esterel plan dispatcher for temporal plans with concurrency. Both dispatchers require as input a complete plan produced offline. For stochastic plan execution with RDDL, a third plan dispatcher was designed and implemented that allows the use of online planners (Figure 3: Nodes *Planner Interface* and *RDDL Plan Dispatch*). The online plan dispatcher interleaves plan execution and computation, removing the need of computing an offline plan and replanning when an action fails.

The online dispatcher uses the RDDL Client/Server protocol, also used by the competition server for the IPPC. In each round, the dispatcher obtains the world's state from the Knowledge Base and sends it to the planner, which returns the actions to be executed in the next time step. This process is repeated until the planner has reached the horizon defined in the instance file, in which case the planning process can be repeated when the task is not yet finished. With this dispatcher, any RDDL planner that uses the RDDL Client/Server protocol can be used with ROSPlan with no extra effort.

4.1 Action Execution with Non-deterministic Effects

A robotic system interacting with the real world must keep the symbolic state of the task up to date, based on its sensory inputs. This means updating the Knowledge Base at a fixed rate such that the state is updated before each action is planned and executed. This is crucial in probabilistic planning, as with non-deterministic action outcomes it is not possible to assume that the effects of each action can be applied to the state. Instead, sensing is required to determine which outcome occurred. Therefore, we implemented a new sensing interface (Figure 3: *Sensing Interface*) that allows the definition of "sensed predicates and functions", which are those whose values are obtained from sensor data.

The sensing interface automatically obtains the sensor data, processes it based on a minimal code definition, and updates the Knowledge Base accordingly at a fixed rate. At the same time, the KB is updated to include the information regarding which propositions are sensed or not, such that effects on the sensed propositions are not automatically applied when an action is executed. The sensed predicates are defined in a configuration file in which is specified: (1) the predicate label; (2) the parameters of the predicate which can be instantiated, and those which are fixed; (3) the sensor containing the required data, expressed as a ROS topic or service; (4) the message type of the data; (5) a single line of code whose result will be the value assigned to the predicate in the KB. Here we show an example of this configuration for a predicate:

```
1. docked:
2.    - params kenny
3.    - /mobile_base/sensors/core
4.    - kobuki_msgs/SensorState
5.    - msg.charger != msg.DISCHARGING
```

This configuration shows (line 1) the name of the predicate, *docked*; (line 2) that the single parameter of the predicate is fixed, so that this configuration is sensing the value of the proposition `(docked kenny)`; (line 3 and 4) the ROS topic to which the sensing interface will subscribe and the message type; and (line 5) a single line of code that returns a Boolean result to be assigned to the proposition.

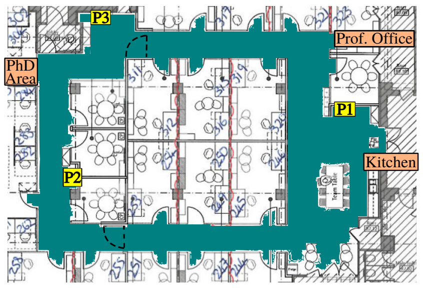
If a more complex processing needs to be done, the interface can be linked with another file containing the implementation for each predicate, in which any kind of program can be defined in order to process the sensor data.

5 Example System and Scenario

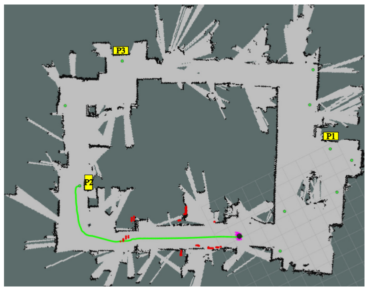
We have used the RDDL nodes in our example scenario, using the system architecture shown in Figure 3. In this system, the RDDL Knowledge Base loads the RDDL domain and initial state. The Problem Interface requests the domain and state information to generate a RDDL problem instance. The Planner Interface and RDDL Plan Dispatch communicate through the IPPC server interface, as described above, suggesting and dispatching actions. The sensing interface is also being used to instantiate the predicates based on sensor data and update the state accordingly.

To demonstrate the effectiveness of the developed framework, we have tested it in a scenario in which a mobile robot fetches printed documents in a large office building. This scenario involves a high degree of uncertainty, since the environment is dynamic and humans can obstruct the corridors and printers. The scenario also involves human-robot interaction, which is intrinsically uncertain.

Scenario description The robot operates in a single-floor office environment with 16 offices shown in Figure 4. There are three printers distributed along the corridor. The robot can trigger printing on any of these printers when a request is made. Since the mobile robot is not equipped with an arm, the robot can request human assistance to place the papers onto its tray. There are many employees working in this area, and the corridor is usually dynamic. The robot relies on the fact that someone will pass by and assist the robot upon request. However, it can happen that there is no one at the printer and the robot has to wait or go to another printer. Once the documents are on the carrier, the robot brings them to the person who made request. It is important to note that printers can be occupied, in which case the robot will have to wait. Moreover, the



(a)



(b)

Fig. 4. (a) The layout of office environment where the robot is operating. The corridor is marked with the green color and printers are marked with yellow boxes. The orange boxes denote potential goal destinations. (b) A screenshot of the visualization tool RViz taken while performing experiments. It shows the map of the corridor and a green line indicating the robot's current path.

robot will know whether there is somebody there to assist or if the printer is busy until it has arrived to the printer. Figure 1 shows an example of the scenario.

This scenario could be well-suited to be modeled as a Partially Observable Markov Decision Process (POMDP), as there are fluents that cannot be known until observed, such as the presence or absence of people near the printer. Also, it could be modeled as an Stochastic Shortest Path (SSP) problem, given that the scenario is goal-oriented in that the robot has to deliver the printed papers to a specific location. However, given the lack of available out-of-the-box solvers for both POMDPs and SSPs, we have modeled the problem as an MDP where a positive reward is given only once the goal is reached.

5.1 Print-fetching domain

In order to run the scenario on both PDDL and RDDL planners, a domain model has to be written in each language[6]. Care must be taken to ensure that the state transition in both domains remains identical, with the exception of probabilistic effects. While the RDDLSim software used to run the IPPC includes an automatic translation from RDDL to a subset of PPDDL, to properly determinize the domain we performed this translation by hand. In future work we intend to investigate the prospect of using the Knowledge Base to perform this determinization automatically.

RDDL domain description The print-fetching domain in RDDL is made of seven action fluents: one for moving (*goto_waypoint*), two actions for interacting with the user and asking him/her to load or take the printed papers, two for waiting for the user to do it, and the ones for docking and undocking the robot to the charging station. A fluent named *goal* is used to specify the goal condition, such that the final reward is given only once the goal is reached, thus simulating a goal-oriented MDP. In the print-fetching domain the goal is to deliver the printed papers to a specific location. The domain has two stochastic fluents, both sampled from a Bernoulli distribution. One represents whether there is somebody to help the robot in one location, and the second specifies whether a printer is being used or not, being the parameter of the Bernoulli distribution dependant on the location. Finally, the reward function provides a positive reward when the goal is reached and the robot is docked, and then some penalizations, considered as costs, for moving (weighted by the distance of the moving action), waiting in a printer where there is nobody to help, and waiting in a printer which is busy.

6 Experiments

In our experiments we used a mobile robot (TurtleBot 2). The robot is equipped with a Kinect sensor which is used for both mapping and navigation [11]. Experiments were run in a real-world office environment where people were performing their regular daily activities. Therefore, corridors were crowded and the robot had to avoid obstacles while performing the task. All actions used in the scenario were implemented, apart for the detection of paper placement and human presence perception, which were simulated. An implementation of these actions is not in the scope of this paper.

[6] Both PDDL and RDDL domains can be found here: `https://github.com/m312z/KCL-Turtlebot/tree/master/domains`

Printer	Events	Probability of the event
P1	Occupancy	0.5
P1	Nearby person	0.9
P2	Occupancy	0.2
P2	Nearby person	0.4
P3	Occupancy	0.8
P3	Nearby person	0.5

Table 1. Prior probabilities of events in the experimental setup. The same values are used in the problem definition of RDDL.

Experiments	Start position	Delivery goal	Printer	Printers occupancy	Nearby person
1	Prof. Office	PhD Area	P1	free	yes
			P2	free	no
			P3	free	no
2	PhD Area	Kitchen	P1	free	yes
			P2	free	yes
			P3	busy	yes
3	Docking station	Prof. Office	P1	busy	yes
			P2	free	no
			P3	busy	yes

Table 2. Experimental setups. For each setup and planning approach we run 5 tests.

We tested the system architecture shown in Figure 3 using the probabilistic planner PROST [16] and compared with the default ROSPlan system using the PDDL 2.1 planner Metric-FF [13]. The goal for both planners is to deliver the printed papers in the shortest time. There were two sources of uncertainty in the scenario whose prior probabilities were modeled in the RDDL domain: (1) the presence of people near the printer and (2) the occupancy of the printer. The values are given in Table 1. When using the deterministic planner (Metric-FF), the system replanned in the case of an action failure.

6.1 Results

We performed three different real-world robotic experiments which represented three situations obtained by sampling the events of person near the printer and occupancy of the printer. These experiments are described in Table 2. For each experiment we applied both planning approaches in five executions. A fourth experiment has been simulated.

As a measure of effectiveness we compare total time of execution, time of planning and robot travel distance. To measure execution time, we measure from the start of planning until the robot completes the task. To measure planning time: (1) in the case of Metric-FF replanning can be performed several times, so the total planning time is the sum of these planning episodes; (2) in the case of PROST, planning is performed before each action is taken so total planning time is the sum of the time to produce each action. The travel distance is the length of the path that the robot traveled.

The results of all three experiments are shown in Figure 5. Experiment 1 demonstrates the advantage of probabilistic planning. In this set up, the robot can only succeed in printer $P1$, though when only the traveled distance is considered, $P3$ would be the best option. In order to minimize the expected duration of the plan, the Metric-FF planner chose to visit printers $P2$ and $P3$ without taking probabilities of events into account. As these printers were empty, the plan execution failed and replanning was performed both times, to finally succeed when visiting $P1$. On average, the Metric-FF planner had to replan 4 times in each test run of this experiment. In contrast, the probabilistic approach attempted to use printer $P1$ first[7].

Experiment 2 shows a simple case where conditions are optimal for a deterministic planner (no unexpected effects). In this case, $P1$ and $P2$ are the best option to select. As expected, the Metric-FF planner did not have to replan at all, as the best solution was the one selected in the first attempt. Therefore, it exhibits a shorter planning and execution

[7] A video demonstration of this setup can be found in `https://youtu.be/aozTz4Ex7PI`

time than the probabilistic planning approach. The distance is still approximately the same, because only in one case did PROST not find the optimal solution.

Experiment 3 shows a case where the available printers are busy, therefore forcing the robot to either wait for the printer to become available or to try another printer. In this case, we simulate the printer to be busy for one action execution. Therefore the printer will become available if the robot waits until a timeout and checks again, or if the robot goes to another location and comes back to a visited printer which was busy. The observed behavior for the deterministic planner in this case was to visit the closest printer $P1$, which was busy, then visit printer $P2$, which was empty, the printer $P3$, which is also busy, to finally succeed at $P1$. In contrast, the stochastic planner went to printer $P1$, waited for it until timeout, and then waited again, obtaining the papers in this second step. This behavior was obtained due to the planner having the certainty of eventually having someone to help at printer $P1$, though there was uncertainty of succeeding if other printers were visited.

The standard deviation (σ) in distance and execution time is small for PDDL, and large for RDDL. This is because the deterministic planner always chooses the plan that is optimal in time and distance, and in fact the σ comes only from real-world execution. The variance seen in PDDL is due to the navigation system and person interaction. In contrast, PROST produces different plans depending upon the probabilities of events, which can vary greatly in execution time and distance travelled. The σ in planning time is greater for the PDDL planner. This is due to the impact of the replanning attempts.

A final simulated experiment has been performed to further show the effects of the probabilities in the planning scenario we proposed. The setup for this experiment was the robot starting at the *PhD Area*, and the delivery goal was the *Professor's office*. For this experiment, 500 executions with both the deterministic planner and the stochastic

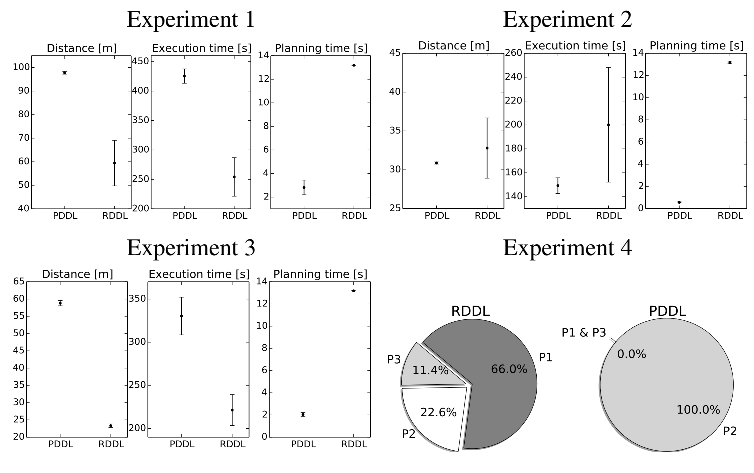


Fig. 5. Experimental results, showing mean values with standard deviations of the robot travel distance, test execution time and planning time for the first 3 experiments. In each experiment, 5 tests were performed for each approach. In Experiment 4, 500 tests were made in simulation. Experimental results show the distribution of the first printer selected across all plans and planner.

one are carried out, and we take into account only the action of the plan that shows the first chosen printer. As it can be seen in the results from Figure 5, the deterministic planner always chose to go to $P2$, which is the one providing shortest travel distance. In contrast, the stochastic planner has different choices, leading to a distribution that resembles the one shown in Table 1, selecting to visit most of the times $P1$, then $P2$ and finally $P3$. Therefore, given that $P2$ is less likely to have people around to help the robot, the deterministic planner is more prone to fail in such setup.

7 Discussions and Conclusions

The focus of this systems paper was to describe the integration of probabilistic planning into ROSPlan, and to demonstrate the execution of probabilistic plans in real-time robotics scenarios. This has involved the implementation of RDDL models into the ROSPlan KB, and an online dispatcher that uses the RDDL Client/Server protocol.

This paper is not intended to make a comparison of deterministic vs. probabilistic approaches. Our experiments show that both approaches have advantages, and a more thorough discussion can be found in [20]. There are many factors that determine which planning approach is better-suited to the domain and problem. For example, whether the domain is probabilistically interesting and whether probabilities are known. Also, whether or not it is necessary to have an optimal plan, or that from a given initial state the same plan is always generated for execution.

Although the use of a probabilistic planner may result in shorter paths and faster plan execution, from the perspective of domain modelling we found it was more intuitive to use an action-oriented language. Another element to take into account is that, while the handling of uncertainties by means of probabilistic planning can be useful in robotics and real-world scenarios, those probabilities must be coherent with the real-world. Such probabilities are often hard to obtain or estimate, and will usually need some kind of learning or adaptation to the real world.

As a contribution of this work, it is possible to combine both of these approaches in ROSPlan. By integrating RDDL into the ROSPlan framework, it is now straightforward to use both PDDL and RDDL planners in a single system. This means a robotic task can be divided into subtasks, in which some are probabilistic and the others deterministic. Moreover, as the online dispatcher conforms to the RDDL Client/Server protocol used in the IPPC, a wide choice of probabilistic planners is made available.

References

1. Atrash, A., Koenig, S.: Probabilistic planning for behavior-based robots. In: FLAIRS. pp. 531–535 (2001)
2. Bonasso, R.P., Firby, R.J., Gat, E., Kortenkamp, D., Miller, D.P., Slack, M.G.: Experiences with an architecture for intelligent, reactive agents. Journal of Experimental & Theoretical Artificial Intelligence **9**(2-3), 237–256 (1997)
3. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research **11**, 1–94 (1999)
4. Buksz, R.D., Cashmore, M., Krarup, B., Magazzeni, D., Ridder, B.C.: Strategic-tactical planning for autonomous underwater vehicles over long horizons. In: IROS (2018)

5. Canal, G., Alenyà, G., Torras, C.: Adapting robot task planning to user preferences: an assistive shoe dressing example. Autonomous Robots pp. 1–14 (2018)
6. Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., Carreras, M.: Rosplan: Planning in the robot operating system. In: ICAPS (2015)
7. Celorrio, S.J., Fernández, F., Borrajo, D.: The PELA architecture: Integrating planning and learning to improve execution. In: AAAI (2008)
8. Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Computational intelligence **5**(2), 142–150 (1989)
9. Fox, M., Long, D.: PDDL2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research **20**, 61–124 (2003)
10. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Elsevier (2004)
11. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Transactions on Robotics **23**(1), 34–46 (2007)
12. Hoey, J., Von Bertoldi, A., Poupart, P., Mihailidis, A.: Assisting persons with dementia during handwashing using a partially observable markov decision process. Vision Systems **65**, 66 (2007)
13. Hoffmann, J.: The Metric-FF planning system: Translating “ignoring delete lists” to numeric state variables. Journal of Artificial Intelligence Research **20**, 291–341 (2003)
14. Hsiao, K., Kaelbling, L.P., Lozano-Perez, T.: Grasping POMDPs. In: ICRA (2007)
15. Iocchi, L., Jeanpierre, L., Lázaro, M.T., Mouaddib, A.I.: A practical framework for robust decision-theoretic planning and execution for service robots. In: ICAPS. pp. 486–494 (2016)
16. Keller, T., Eyerich, P.: PROST: Probabilistic planning based on UCT. In: ICAPS (2012)
17. Kolobov, A., Dai, P., Mausam, M., Weld, D.S.: Reverse iterative deepening for finite-horizon MDPs with large branching factors. In: ICAPS (2012)
18. Krivic, S., Cashmore, M., Magazzeni, D., Ridder, B., Szedmak, S., Piater, J.: Decreasing Uncertainty in Planning with State Prediction. In: IJCAI. pp. 2032–2038 (8 2017)
19. Kushmerick, N., Hanks, S., Weld, D.S.: An algorithm for probabilistic planning. Artificial Intelligence **76**(1-2), 239–286 (1995)
20. Little, I., Thiebaux, S.: Probabilistic planning vs replanning. In: ICAPS Workshop on Planning Competitions: Past, Present, and Future (2007)
21. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: ICML. pp. 157–163 (1994)
22. Martínez, D., Alenyà, G., Ribeiro, T., Inoue, K., Torras, C.: Relational reinforcement learning for planning with exogenous effects. Journal of Machine Learning Research **18**(1), 2689–2732 (2017)
23. Martínez, D., Alenyà, G., Torras, C.: Relational reinforcement learning with guided demonstrations. Artificial Intelligence **247**, 295–312 (2017)
24. Pacchierotti, E., Christensen, H.I., Jensfelt, P.: Design of an office-guide robot for social interaction studies. In: IROS. pp. 4965–4970 (2006)
25. Sanner, S.: Relational dynamic influence diagram language (RDDL): Language description. Unpublished Manuscript (2010)
26. Smith, B.D., Rajan, K., Muscettola, N.: Knowledge acquisition for the onboard planner of an autonomous spacecraft. In: EKAW (1997)
27. Smith, T., Simmons, R.: Probabilistic planning for robotic exploration. Ph.D. thesis, Carnegie Mellon University, The Robotics Institute (2007)
28. Veloso, M., Biswas, J., Coltin, B., Rosenthal, S., Kollar, T., Mericli, C., Samadi, M., Brandão, S., Ventura, R.: Cobots: Collaborative robots servicing multi-floor buildings. In: IROS. pp. 5446–5447 (2012)
29. Yoon, S.W., Fern, A., Givan, R.: FF-Replan: A baseline for probabilistic planning. In: ICAPS. pp. 352–359 (2007)