UAV MISSION PLANNING AND EXECUTION VIA NON-DETERMINISTIC AI PLANNING ON ROS

Vinícius Veloso Eleuterio Nogueira*, Luiz Edival de Souza†

* *Universidade Federal de Itajubá, av. B P S, 1303 - Pinheirinho, Itajubá- MG, Brazil*

Emails: nogueira.vinicius@hotmail.com, edival@unifei.edu.br

Abstract— Unmanned aerial vehicles (UAVs) have been attracting civilian attention by its increasing number of applications and decreasing cost. The vast majority of current UAV tasks still have a low degree of autonomy, which can lead to lack of efficiency, safety, practicality and viability. This paper proposes using Probabilistic AI Planning to improve autonomy and robustness in UAV applications. Our main contribution is a framework that integrates Prob-PRP planner on ROS systems and enables the visualization, execution and monitoring of its plans using Finite State Machine (FSM). On this article, we describe our framework and apply it for high-level mission control on a simulated autonomous UAV.

Keywords— Automated Planning, Artificial Intelligence, Robotics Sytems, Unmanned Aerial Vehicles.

Resumo— Os veículos aéreos não tripulados (UAV) têm atraído a atenção das pessoas civis pelo seu crescente número de aplicações e pela redução de custos. A grande maioria das tarefas atuais do UAV ainda possui um baixo grau de autonomia, o que pode levar à falta de eficiência, segurança, viabilidade e praticidade. Este artigo propõe o uso de planejamento de AI probabilístico para melhorar a autonomia e robustez nas aplicações de UAV. Nossa principal contribuição é uma estrutura que integra o planejador Prob-PRP em sistemas ROS e permite a visualização, execução e monitoramento de seus planos usando a Máquina de Estados Finitos (FSM). Neste artigo, descrevemos o nosso quadro e aplicamos o controle de missão de alto nível em um UAV autônomo simulado.

Palavras-chave— Planejamento Automático, Inteligência Artificial, Sistemas Robóticos, Veículos Aéreos Não Tripulados

1 INTRODUCTION

The development of unmanned aerial vehicles (UAVs) began in applications involving military operations (Turner et al., 2012) during the Vietnam War or Cold War (Watts et al., 2012) and became popular for civilian use with the rise of multicopters. These vehicles gained relevance mainly due to their mechanical simplicity, low cost, user-friendliness, and safety (Cutler, 2012).

Nowadays, UAVs can be found in a wide number of civil applications. These applications include: aerial photography; agriculture; mapping; search and rescue; disaster management; monitoring; border patrol; inspection; environmental control; remote sensing and wireless relay. On top of that, there are still a large of number of studies that aims to increase its capabilities. Future applications may require higher level of intelligence from the system as they tend to become more complex. Most of the studies involving these types of vehicles still focus on perception and control strategies(Bernardini et al., 2014). Moreover, high levels of autonomy will require the system to act deliberately.

A promising field within Artificial Inteligence that could provide intelligence to the system is *Automated Planning*, also called AI Planning. Automated Planning is the reasoning side of acting (Ghallab et al., 2004) that relies on a model of the agent and its environment to produce a plan that should guide an agent to reach its objectives. This model can be domain specific planning (Mersheeva and Friedrich, 2015)(Ghamry et al., 2016)(Lee and Morrison, 2015) (Albore et al., 2015) or domain-independent (Bernardini et al., 2014) (Munoz-Morera et al., 2015). Nowadays, domain specific solutions outperform domain independent plans, but they need much modeling effort which requires much specialized knowledge and limits deliberative capabilities to specific areas. On the other hand, domain independent strategies attract more research interest, and had received much contribution in the past few years. These contributions envolve algorithms that either increase expressivity or decrease the complexity of domain independent approach.

UAVs are naturally unstable systems that have high sensitivity to disturbances, such as wind and rain. In addition, the limitation of power sources makes the energy consumption of these vehicles crucial to meet their objectives (Sydney et al., 2013). Therefore, considering the environmental uncertainty during task planning show to be necessary in this type of vehicles, once that deterministic approaches do not avoid dead-ends. We selected Prob-PRP(Camacho et al., 2015) non-deterministic solver that aims to increase the plan probability of success. It is a goal-oriented and domain-independent solver that uses PPDDL model as input.

The combination of Automated Planning and aerial vehicles was already proposed in (Cantoni et al., 2011), where it was studied classical planning on a fire fighter scenario and simulated it on X-Plane®. Another similar research was made

by (Bernardini et al., 2014) in aerial surveillance planning for searching-and-tracking operations. Automated Planning is also applied commonly to other types of robotic applications (Quintero et al., 2011) (Crosby et al., 2017). As automated planning became the standard tool for domain-independent problem solving and Robotic Operating System (ROS) for robotics applications they were first merged in (Cashmore et al., 2015) resulting in ROSPlan. In its first version, this package implemented a classical planner and was used in an application using autonomous underwater vehicles. Later, it has been extended to support contingency plans, using Contingent-FF (Sanelli et al., 2017). While contingency plans are able to handle eventualities, monotonicity of knowledge prevents construction of cycles which can be required in some domains.

ROS (Quigley et al., 2009) is popular modular framework that allows the integration of libraries intended for robotic systems. It provides hardware abstraction and communication to interface different robotic application. The implementation in this platform is convenient once it already possess many state-of-the-art algorithm from scientific collaborative work.

Our approach was implemented in ROS using both Python and C++. Our presented library is based on ROSPlan (Cashmore et al., 2015) and conditional planning in (Sanelli et al., 2017). Our framework contributes to the community by incorporating PRP and allowing its plan execution, visualization and supervision on ROS systems. Its purpose is to designate high-level tasks that guides an UAV to its goal according to the outcomes of its actions using a FSM controller. We call this framework *DOTPlan*.

This paper first provides some preliminaries on Finite State Machines and their relation to our work in section 2. On section 3, we do a overview on the system, describing its architecture, features and purpose. Following, we apply Prob-PRP on UAV delivery domain using our framework and discuss the results.Following, we present the integration with simulated application and compare our work with others in section 5. Future work and conclusion are discussed in section 6.

2 Preliminaries

Finite State Machines (FSMs) are a computation model represented by finite number of states and transitions. A deterministic finite automaton (DFA) is defined in (Hopcroft et al., 2006) as a tuple $(Q, \Sigma, \delta, q0, F)$, where Q is a finite set of states; Σ is a finite set of symbols, called alphabet; $\delta : Q$ x $\Sigma \rightarrow Q$ is a deterministic transition function; $q0 \in Q$ is the initial state; $F \subseteq Q$ is a set of goal states. It is a abstract machine that receives a set of inputs alphabet symbols, called string, and process them sequentially according to the transition functions. After processing all inputs, if it ends on one of the goal states, the machine accepts the string, if not it rejects it. A FSM can be graphically represented as graph in which the nodes are equivalent to the states and the transition to the edges. Figure 1 shows a DFA with $(\{q0, q1, q2, q3, q4\}, \{a, b, c\}, \delta, q0, \{q3\})$, with δ being described by the graph edges. This automaton will accept any string that have three *a*s and no *c*s.

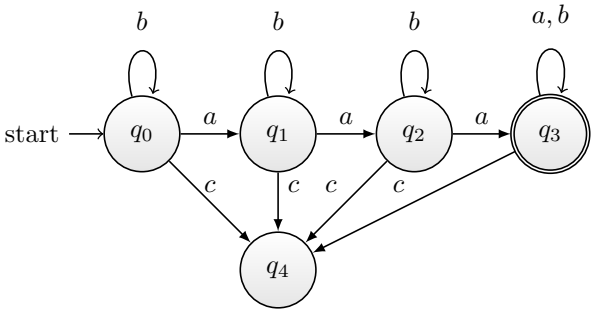


Figure 1: Deterministic Finite Automata(DFA)

2.1 From Non-Deterministic Problem to FSM

A non-deterministic planning problem is described as $P = (S, I, A, T, G)$ in (Kissmann and Edelkamp, 2009), where S is a set of states; I is the initial state; $A(s)$ is a set of applicable actions; G is the goal state; T is a non-deterministic transition relation $T(s, a, s)$ that represents the probability of occurring the outcome, o, when applying action a in the state s. The non-deterministic planning problem is determined when the probabilistic planner parses I,G and other domain knowledge provided in PPDDL format and infers A, T, S. The planner task is to calculate a policy $\Pi(s) \rightarrow a$ that maps a state s into an action a that will guide s to G.

A FSM $(Q, \Sigma, \delta, q0, F)$ can be produced from the policy, if we let Q be a set of states, where a state s is characterized by a set of facts; Σ be a set of all possible determinized actions, ao; δ the transition function that relates s x $ao \rightarrow s$; q0 the initial state, F the state that contains all goal facts. FSM can be created by simulating the policy execution from the initial state to the goal state for every possible outcome. The PRP planner authors (Sardina and D'Ippolito, 2015) made their source code available in which contained a script used for their validation. This script generates a FSM from policies in FOND domains and outputs it in a DOT format. The FSM generation method used by this script is very similar to the algorithm described in (Iocchi et al., 2016) to convert polices into conditional plans.

3 System Overview

The aim of the proposed paper is to introduce a new framework which aims to allow execution of probabilistic plans with ROS and to apply it to a common UAV delivery domain problem, represented in figure 2. This figure shows that the system is composed by modules: planning, supervision, execution and visualization. In this section we describe the framework for executing probabilistic plans via Finite State Machine representation.

3.1 AI Planning

The execution of the planning module is fundamental once it gives the information that the other modules needs to operate. The Planning System is off-line and should output a FSM in Graphviz's Dot format. In our application, the Prob-PRP was applied as the third-party planner resulting in a policy that was converted using algorithm 1. Additional information is also collected during FSM generation, this data should help the system operator during system supervision. FSMs are often cited in literature as a simple and compact controller. This controller is often used for sequential action coordination. While (Sardina and D'Ippolito, 2015) use a FSM as validation tool, we apply it as a plan controller.

3.2 DotViz

DotViz is the visualization module of DOTPlan, it displays the conditional plan graph, highlights and centers the execution's current state. It facilitates the analysis and comprehension of the current plan. This module auto arranges of the graph's nodes and edge in a way which is both compact and clear.

DOT format is a widely used output format for graph representation. While it supports some fancy tools, the basic usage requires only minimal data overhead. Figure 3a shows the DOT of the graph displayed in figure 3b. In figure 3a, it can be noted that the nodes position can be left unspecified. It happens because softwares that reads this format, uses Graphviz library. Graphviz's coordinates assignments enables a clean auto generated graph visualization, what justify its usage in a wide number of application. In this library the problem of arranging is solved as a cost minimization task. It is already present in core ROS package as rqt_graph and rq_tf_tree, wrapped by Pydot. Also, there is a wide variety of libraries available in many different programming languages to automatic create a graph in DOT format, which its increases flexibility, convenience and support. In fact, all diagrams in this paper (except figure 1) were designed in DOT.

Xdot (Fonseca, 2017) is an open source software written in Python that displays DOT files using Graphviz library on a GTK3 user interface. The appliance of Xdot becomes convenient, since it uses Graphviz in a lower level and has some interesting built-in functions like: node_search and animate_to_position. This software was modified to be incorporated on ROS using gobject thread synchronization and called XDotViz. This software is able to show the FSM, auto highlight states and animate through the graph based on information exchange in either topics or services. Figure 4 represents the XDotViz interaction within the framework. Xdot natively does not save any information, therefore it has to arrange the whole graph every time the dot file is opened. In that matter, a new function was implemented to save and load nodes positions in a intermediate file to speed up the program start-up of pre opened files. This feature is useful once that a complex plan can result in a large graph, it could take sometime to auto arrange nodes. XDotViz main purpose is to monitoring the execution of the FSM. The system was implemented in a way that it works independently from XDotViz.

3.3 DotActionServer

DotActionServer is the core module for execution the computed plan. It coordinates the agent high-level actions to the desired goal. It works by passing action information to low-level control and receives its feedback to compute the next action. This information includes possible outcomes, action name, action parameters. The application system should then execute the action and return which was the resulting outcome.

By looking at figure 3a, it can be observed that a DOT form has a simple and efficient structured format. Based on this, the DOT form was defined as the standard input format, not only for monitoring but also for execution. In order to obtain its DOT file the system calls the planner and the converter, which is responsible to transform the plan/policy into a FSM in DOT. This DOT file is referenced to XDotViz and parsed into the plan executor. In this framework, we adopt the following convention to express a FSM in DOT: the node id must be a unique integer; the node name can be arbitrary, with exception to brackets and dashes; *action[outcomes effects]* must be the format of the edge name; # is a reserved character to represents either blank or none.

The action dispatching algorithm parses the DOT file, and extract the following relations: $(s \rightarrow (state\ name))$, $(s \rightarrow a)$, $(s \rightarrow O(s))$, and $(s \text{ x } o \rightarrow s)$. The action outcomes may contain many shared effects, the solution was to express it to its exclusive effects that differs it from the other outcomes. This operation makes the graph

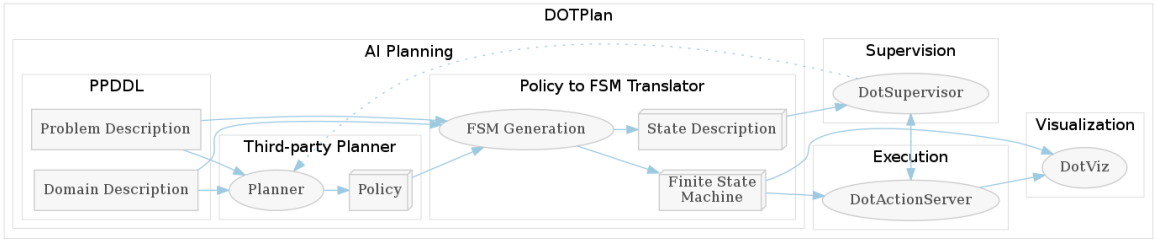
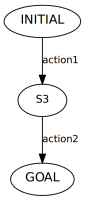


Figure 2: DotPlan System Overview

```
digraph  {
node [label="\N"];
1 [label=INITIAL];
3 [label=S3];
1 -> 3 [key=0,
label="action1"];
4 [label=GOAL];
3 -> 4 [key=0,
label="action2"];
}
```

(a) DOT



(b) Graph

Figure 3: DOT and its corresponding graph

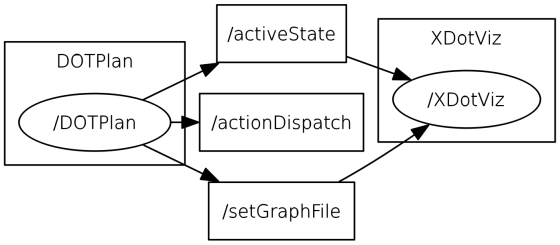


Figure 4: XDotViz interaction with the framework

visualization clearer and eases the outcome verification. The execution of this controller is very straightforward, the controller should keep track of the FSM current state and dispatch the action specified in its transitions. When the action server is activated, it dispatches the initial action and dispatches the consecutive action based on their feedback.

Algorithm **??** shows the procedure to dispatch actions based on their feedback. A plan fails if: the action did not succeed; the action server goes out of synchrony with the action executor; the outcome is not present in the FSM. If a plan doesn't fail it will be able to retrieve the necessary information from the relations until it achieves the goal state.

3.4 DotSupervisor

This framework also includes an interactive graphical user interface, that is used for information output and plan interaction. Some of the information displayed is gathered during the FSM generation. This data includes the active state, action server status, current action, facts s and key facts

procedure ActionFB
Require: $actState, curOut, curAct, Succeed$
  if $Succeed = false$ **then**
    $planStat \leftarrow failed$
    return
  if checkFB$(actState, Out, Act)$=false **then**
    $planStat \leftarrow failed$
    return
  $actState \leftarrow$ progress$(curOut)$
  if stateIsGoal$(actState)$ = true **then**
    $planStat \leftarrow concluded$
    return
  $curAct \leftarrow$ getStateAction$(actState)$
  $stateOuts \leftarrow$ getStateOutcomes$(actState)$
  publishAction$(actState, curAct, stateOuts)$
  publishActiveState(getStateName(s))

s' of the active state. This interface also enables the user to: (re)plan from PPDDL files, to reload the action server DOT file, send action feedback.

4 Prob-PRP in UAV domain

Planner for Relevant Policies (PRP) is the state-of-the-art FOND planner. Prob-PRP is an extended version of PRP to solve probabilistic problems. An interesting characteristic of Prob-PRP is that it guarantees to avoid avoidable dead-end state. This solver relays on finding strong cyclic solution for the problem. A strong cyclic solution is defined as a solution that reaches the goal after an infinite number of actions. One issue is that this planner does not guarantee a solution in the presence of unavoidable dead ends.

A fragment of our domain is displayed in figure 5. The UAV domain is a discrete probabilistic problem that models the task of using a single aerial vehicle for package delivery. This domain considers that routes between locations of interest are discretized with equidistant waypoints. It contains uncertainty in fuel consumption while moving between waypoints. This uncertainty could be related either to weather conditions, necessaries maneuvers or discretization error. The actions $land$ and $take-off$ also consumes fuel but does considers uncertainty on its usage. In this model, the fuel resource is discretized into fuel levels. For

```
(:domain(quaddelivery)
(:types pack way fuellvl - object)
(:predicates
  (deliverat ?pkg - pack ?wp - way)
  (stationat ?wp -way)
  (pkgat ?pkg - pack ?wp - way)
  (quadat ?wp - way)
  (inquad ?pkg - pack)
  (delivered ?pkg - pack)
  (connected ?wp1 ?wp2 - way)
  (quadfuel ?fuel - fuellvl)
  (fuelmap ?lowlvl ?highlvl fuellvl)
  (maximumfuel ?maxfuel - fuellvl)
  (onair)
   (grounded))
(:action move-to
  :parameters (?wp1 ?wp2 - way
               ?f1 ?f2 ?f3 - fuellvl)
  :precondition (and(quadat ?wp1)
    (connected ?wp1 ?wp2)
    (quadfuel ?f3)
    (fuelmap ?f1 ?f2)
    (fuelmap ?f2 ?f3)
    (onair))
  :effect (and (not (quadat ?wp1))
    (quadat ?wp2)
    (not(quadfuel ?f3))
    (probabilistic
                0.6 (quadfuel ?f2)
                0.4 (quadfuel ?f1))))
(:action land (?f1 ?f2 - fuellvl)
(:action takeoff(?f1 ?f2 - fuellvl))
(:action load(?pkg - pack
              ?wp - way))
(:action deliver(?pkg - pack
                 ?wp ?wp2 - way))
(:action refuel(?wp - way
                ?f1 ?f2-fuellvl))
```

Figure 5: Fragment of the UAV Delivery Domain in PPDDL

this problem we suppose that the system holds full observability.

After experimenting with multiple different problem file and domain variations, we could analyze the behavior of Prob-PRP in our domain. When we let the UAV refuel infinite number of times some aspects can be observed: the planner always avoided dead ends when possible; the most probable outcome tends to have the shortest path; when the action results in a least probable outcome; it forces a cycle; it cycles the plan even when the worst set of outcomes would lead to the goal. Bringing these characteristics to the analyzed domain several points can be remarked. For the sake of state reduction and problem simplification, it forces cycles in its plan, in its plan, when is possible to make the choice infinite times, called *strong cycles*. Another problem is when it refuels and takes actions to waste its fuel and return to a state which it visited earlier. In other words, the execution leads back to a state that it previously visited to try to get another outcome. This cycle may increase the plan length and may not be needed to avoid dead ends.

In this planner, the length of the plan is shorten for UAV domain by limiting its number of refuels. By doing that, we take of the guarantee of reaching the goal and permit an increase in the number of states for the FSM. We implicitly impose a heuristic, limiting the mission length by giving it limited fuel. Table 1 shows how the number of allowed refuels affects a domain with 12 waypoints, 10 fuel levels, 2 stations and 3 packages. In this table we can extract the relation between the number of allowed refuel and the probability of success. In that relation the planner percentage of success increases and converges to a value, while the planning time tend to increase in a exponentially. In non-deterministic planning the trade off between success rate and plan length is inevitable, once conservative behavior do not take unnecessary risks. In the experiments, even when the 10% longest plan lengths of the infinite refuels scenario are discarded, the average still 112 actions. This shows that the plan is taking unnecessary actions.

R	PT (s)	S	P	L
2	0.44	0	0	0
3	9.84	80	0.448	18.42
4	35.34	163	0.911	34.25
5	53.26	256	0.966	38.98
6	59.9	258	0.968	43.85
7	102.7	388	0.973	45.02
∞	0.82	150	1	123.04

Table 1: Policy caracteristics by number of allowed refuels in Prob-PRP. R represents the maximum refuels allowed; PT is the planning time, S is the number of the states in the FSM, P is the probability of reaching the goal state; L is the average number of actions to achieve the objective

This approach breaks the cycles and makes a tree that is more likely to converge in the sub-goals and refueling. Even without *strong cycles*, Prob-PRP avoided avoidable dead-ends. Despite that this works and give good solutions, this method breaks Prob-PRP proposed method of assuring plan maximum probabilities using *strong cycles*. This leads to no guarantees in terms of performance. It is also evident that any other approach than the strong cyclic takes much more planning time. This solver seems very practical in problems where the most problem outcome is very likely, in fault recovery and in assuring action desired

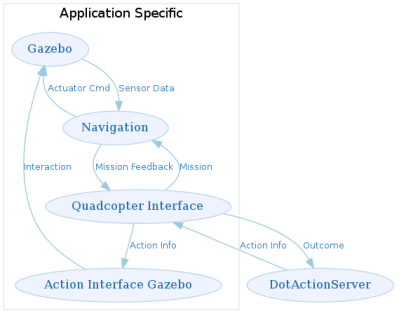


Figure 6: Interfacing DOTPlan's execution with UAV application.

effect(e.g. fire fighting, search and monitoring). Like other model based approaches, the effectiveness of executing this plan will depend on how accurate is the model domain. This planner is a promissing algorithm that is still under development, it is capable of avoiding avoidable dead-ends while being a fast algorithm that gives compact representations.

5 RESULTS

To validate our approach, the proposed framework was implemented in a simulated UAV system using the architecture represented in figure 6. This figures shows how the application interfaces the execution module inside ROS using topics. In this simulation the agent, environment and problem was described in PDDL file format and the actions of the quadrotor are deliberated by the planner and dispatched by the FSM executor. The simulator used for this application was Gazebo, which is the default 3D simulator in ROS. Our platform was the hector quadrotor which is natively compatible with Gazebo, which is equipped by a laser scan, sonar altitude sensor and a RGB camera. In order to keep the rest of the system as simple as possible a 2D navigation stack was applied for moving and other actions interact directly inside gazebo. The resulting system is displayed in figure 7, it exposes the aerial vehicle executing a plan to achieve its objectives.

We compared our framework to the main ROS package for planning and execution, ROSPlan, in its two main variants using POPF and Contingent-FF. This comparison is represented in table 2. All of them use domain-independent planners that accept PDDL structure and present some sort recovery behavior to deal with uncertainty. While ROSPlan-POPF uses online replanning when it encounters uncertainty, ROSPlan-Cont.FF and DOTPlan rely on following a conditional plan. Both replanning and conditional plan need estimation of the current state to guide its plan. But while non-deterministic planners do offline reasoning for different probable sce-

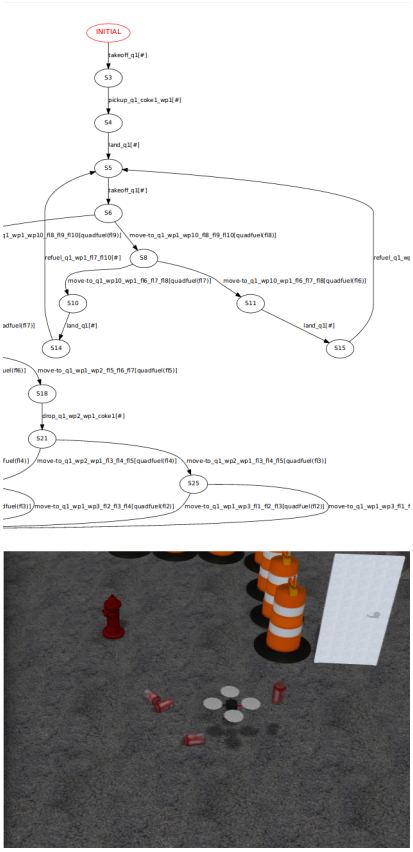


Figure 7: Plan execution in a simulated environment using Gazebo.

narios to avoid dead-ends, replanning does not, which may lead the agent into a dead-end state or repeated interaction in presence of uncertainty (Little et al., 2007). ROSPlan-POPF represents its plan in a list format, which is enough to represent a classical plan since there is only one possible sequence of actions. It is useful to keep track of the agent progress. A problem of visualizing classical plans is that its behavior can be unpredictable if the replanning is triggered. ROSPlan-Cont.FF uses Petri Net Plans (PNP) by interfacing a PNP tool outside ROS called PNPJarp. Petri Net Plans is the application of Petri Nets to represent and execute plans. Its theory also supports concurrent actions and multi-agent systems. We tried to implement our policy in their visualization program, but we had some difficulties with the automatic layout of the states. Also, their PNP generation library doesn't seem to support loops in their policy to PNP translation.

ROSPlan-Cont.FF has a limited perception exogenous events on the environment after it does its observation. This happens because contingent planning assumes that once a fact is unknown it can only changed by an agent action. This is the principle of contingent planning which allows the uncertainty to be compiled in the initial state and provide monotonicity of knowledge. In this way, this system does not support repeated sensing on

	ROSPlan (Popf)	ROSPlan (Cont)	DOTPlan (Prob)
PDDL Support	X	X	X
Recovery Behavior	X	X	X
Off-line Recovery		X	X
Visual Representation	X	X	X
Plan Supervision	X		X
Knowledge Management	X		
Repeated Sensing	X		X
Partial Observability		X	

Table 2: Comparison between ROSPlan using POPF, ROSPlan using Contingent-FF with PNP and DOTPlan using Prob-PRP

the same fact. This doesn't make these technique worst, just not adequate for this kind of problem. In fact, it is able to solve partial observability problems which Prob-PRP and POPF are not. ROSPlan-POPF keeps track of all knowledge using its knowledge base. In this package, sensing actions can occurs infinitely often and influences on how the system estimates its state and update its known facts. Once that the system facts changes, replanning should be triggered. DOTPlan must observe the effects of its actions after every non-deterministic action to choose which branch of its plan that it will dispatch.

6 Conclusion and Future Work

In this paper we presented a framework for executing probabilistic plans with ROS Systems and applied it into the UAV delivery domain. The proposed package is based in the representation of plans with finite state machines (FSM) in DOT format. This system utilizes Prob-PRP as its default probabilist planner but it also supports PRP and PO-PRP, once that they all have the same output format. We applied this solver into the UAV domain and made several considerations about its applicability in UAV systems. Finally, it was compared to other standard planning tools inside ROS, highlighting the possibility of solving probabilistic problems by strong cycles.

While simulated domain might presented only the modeled outcomes, this might not happen in a real-world application. Thereat, it is relevant to incorporate replanning into the framework. This would require a tool for problem generation. Since we already keep track of the state facts, defining the initial state of the new plan shouldn't be a big issue. In the present time, this framework doesn't support concurrent actions. This is a known limitation of doing execution using FSMs. We believe that this could be overcome with an action feedback buffer, similar to Determinist Pushdown Automaton. In the future, this framework could also be merged into other planning frameworks.

References

Albore, A., Peyrard, N., Sabbadin, R. and Teichteil-Königsbuch, F. (2015). An online replanning approach for crop fields mapping with autonomous uavs., *ICAPS*, pp. 259–267.

Bernardini, S., Fox, M. and Long, D. (2014). Planning the behaviour of low-cost quadcopters for surveillance missions., *ICAPS*, Portsmouth, NH, pp. 445–453.

Camacho, A., Muise, C., Ganeshen, A. and McIlraith, S. A. (2015). From fond to probabilistic planning: Guiding search for quality policies, *Workshop on Heuristic Search and Domain Independent Planning, ICAPS*.

Cantoni, L. F., Campos, M. F. and Chaimowicz, L. (2011). Investigacao da linguagem pddl no planejamento de missoes para robËos aéreos, *X SBAI*.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtós, N. and Carreras, M. (2015). Rosplan: Planning in the robot operating system., *ICAPS*, pp. 333–341.

Crosby, M., Petrick, R. P., Toscano, C., Dias, R. C., Rovida, F. and Krüger, V. (2017). Integrating mission, logistics, and task planning for skills-based robot control in industrial kitting applications, *Ceur Workshop Proceedings*, Vol. 1782.

Cutler, M. J. (2012). *Design and control of an autonomous variable-pitch quadrotor helicopter*, PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics.

Fonseca, J. (2017). Xdot.py: An interactive viewer for graphs written in graphviz's dot language., https://github.com/jrfonseca/xdot.py.

Ghallab, M., Nau, D. and Traverso, P. (2004). *Automated Planning: theory and practice*, Elsevier.

Ghamry, K. A., Kamel, M. A. and Zhang, Y. (2016). Cooperative forest monitoring and fire detection using a team of uavs-ugvs, *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, IEEE, pp. 1206–1211.

Hopcroft, J. E., Motwani, R. and Ullman, J. D. (2006). Automata theory, languages, and computation, *International Edition* **24**.

Iocchi, L., Jeanpierre, L., Lazaro, M. T. and Mouaddib, A.-I. (2016). A practical framework for robust decision-theoretic planning and execution for service robots., *ICAPS*, pp. 486–494.

Kissmann, P. and Edelkamp, S. (2009). Solving fully-observable non-deterministic planning problems via translation into a general game, *KI 2009: Advances in Artificial Intelligence* pp. 1–8.

Lee, S. and Morrison, J. R. (2015). Decision support scheduling for maritime search and rescue planning with a system of uavs and fuel service stations, *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, IEEE, pp. 1168–1177.

Little, I., Thiebaux, S. et al. (2007). Probabilistic planning vs. replanning, *ICAPS Workshop on IPC: Past, Present and Future*.

Mersheeva, V. and Friedrich, G. (2015). Multi-uav monitoring with priorities and limited energy resources., *ICAPS*, pp. 347–356.

Munoz-Morera, J., Maza, I., Fernandez-Aguera, C. J., Caballero, F. and Ollero, A. (2015). Assembly planning for the construction of structures with multiple uas equipped with robotic arms, *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, IEEE, pp. 1049–1058.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y. (2009). Ros: an open-source robot operating system, *ICRA workshop on open source software*, Vol. 3, Kobe, p. 5.

Quintero, E., García-Olaya, Á., Borrajo, D. and Fernández, F. (2011). Control of autonomous mobile robots with automated planning, *Journal of Physical Agents*, Citeseer.

Sanelli, V., Cashmore, M., Magazzeni, D. and Iocchi, L. (2017). Short-term human-robot interaction through conditional planning and execution, *ICAPS*.

Sardina, S. and D'Ippolito, N. (2015). Towards fully observable non-deterministic planning as assumption-based automatic synthesis., *IJCAI*, pp. 3200–3206.

Sydney, N., Smyth, B. and Paley, D. A. (2013). Dynamic control of autonomous quadrotor flight in an estimated wind field, *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, IEEE, pp. 3609–3616.

Turner, D., Lucieer, A. and Watson, C. (2012). An automated technique for generating georectified mosaics from ultra-high resolution unmanned aerial vehicle (uav) imagery, based on structure from motion (sfm) point clouds, *Remote Sensing* **4**(5): 1392–1410.

Watts, A. C., Ambrosia, V. G. and Hinkley, E. A. (2012). Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use, *Remote Sensing* **4**(6): 1671–1692.