Generating Plans for Cooperative Connected UAVs

François Bodin, Tristan Charrier, Arthur Queffelec, François Schwarzentruber
Univ Rennes, CNRS, IRISA
francois.bodin@irisa.fr, tristan.charrier@irisa.fr,
arthur.queffelec@irisa.fr, francois.schwarzentruber@ens-rennes.fr

Abstract

We present a tool for graph coverage with a fleet of UAVs. The UAVs must achieve the coverage of an area under the constraint of staying connected with the base, where the mission supervisor starts the plan. With an OpenStreetMap interface, the user is able to choose a specific location at which the mission needs to be generated and to observe the resulting plan being executed.

1 Introduction

In many applications (automated farming, search and rescue, decontamination of a zone, etc.), unmanned autonomous vehicles (UAVs) have to map or to patrol around a *highly constrained* geographical area.

We consider a geographical area, with a launch base and regions of interest to visit, and topological communication constraints. Our goal is to synthesize a cooperative plan for the fleet of UAVs for visiting all the regions of interest at least once, always keeping communication with the base. We suppose that communication may be multi-hop (a UAV may communicate to the base via intermediate UAVs). We call this problem the *connected cooperative coverage problem*. The difficulty resides in the combinatorial when UAVs cooperate to keep communication with the base all along the plan. A geographical area is modeled by a graph.

Our setting is similar to the one proposed in [Teacy *et al.*, 2010] and [Yanmaz, 2012], but we implemented an offline algorithm, that is sound and complete. Connectivity can be maintained by analytic techniques [Giordano *et al.*, 2011] but we need to plan offline in order to capture the combinatorial aspects of a whole geographical area.

Figure 1 shows an execution of an 11-step plan in a graph: the base is the red dot; solid lines represent elementary possible moves between two regions; a dashed line between two regions means that communication is possible between them. At the first step, UAVs are at the base. At the end, all regions are visited. During the execution, UAVs stay connected to the base (dashed lines forming a connected subgraph).

The system is synchronous: all UAVs alternate between moving and visiting regions of interest to perform some tasks. UAVs are stationary while performing tasks and are supervised by humans at the base; so UAVs communicate huge amount of data to the base. Importantly, UAVs do not communicate while moving since no task has to be supervised at that moment. Nodes in the discrete graph are the task locations, plus other intermediate locations for multi-hop. High-speed broadband communications (e.g. laser) are required and do not pass through buildings. So the connectivity issue is not trivial. Indeed, to reach a node, it is also needed to plan for each intermediate UAV. Therefore, the planning tasks for all UAVs are inter-dependent. That is why we start with a simple formulation: synchronism, all nodes are tasks and distances are omitted in the graph.

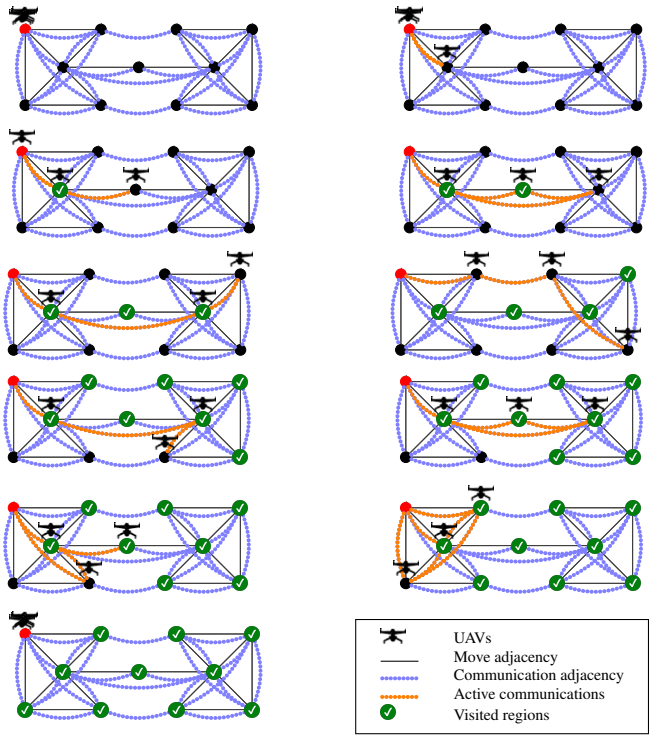

Figure 1: Example of a mission execution

In this paper, we propose a tool for generating plans for multi-UAV systems that should visit all regions of interest while maintaining the connectivity via multi-hop. The tool can be found at the following URL: http://dronestrategy.irisa.fr/

Figure 2: Selection of a zone

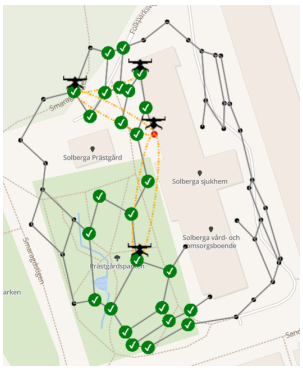

Figure 3: Execution of a plan

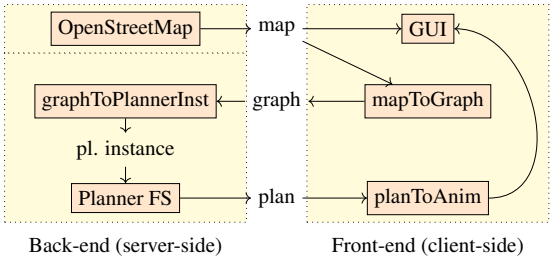

Figure 4: Architecture of the tool

2 Demonstration Outline

The software starts by showing an OpenStreetMap (https://www.openstreetmap.org/) map. The user can choose the number of UAVs and the maximal distance a UAV can travel in one elementary step. Then, the user selects the zone to cover (Figure 2). Once the zone is selected, the software automatically generates a discrete graph, where each node is a relevant region of interest. It avoids to provide the input graph manually. The base is chosen automatically and can be changed by the user. Finally, the software computes and displays a plan for the UAVs to visit the graph while maintaining the connectivity to the base (Figure 3, the base is the red dot).

3 Architecture of the Tool

In this section, we describe the architecture of our tool, as shown in Figure 4.

Front-end The front-end of the tool uses the API MapBox (https://www.mapbox.com/) for OpenStreetMap, which extracts the buildings in the selected zone. Then, mapToGraph computes the graph from the set of buildings with a Delaunay triangulation (the method is inspired from [Jørgensen and Lamarche, 2011]). Communication edges are added whenever two locations are in line of sight. The graph is finally given to the back-end and when the plan is received, it is displayed with the planToAnim function.

Back-end The back-end of the tool takes as input the graph, given to the graphToPlannerInst function. Then, it converts the graph as a classical planning instance in Planning Domain Description Language (PDDL) [McDermott *et al.*, 1998]. Finally, the planning instance is solved with the planner FS (for Functional STRIPS) [Francès *et al.*, 2017]. The returned plan is a sequence of actions such that the connectivity and the coverage constraints are respected.

4 Experimental Results

We tested the tool on various areas, and empirically concluded that the time complexity for a plan search depends on the maximal number of intermediate UAVs N needed for any UAV to communicate with the base. For instance if $N = 0$, any UAV can communicate directly with the base wherever it is. If $N = 1$, every UAV can communicate with the base using at most 1 intermediate UAV. For instance, with the set of buildings of Figure 2 and 3, $N = 3$.

In order to get an idea of the running time, we show some examples with $N = 1$, 2 and 3, with more and more refined graphs. We define d as the maximum distance of a move edge and n as the number of nodes of the generated graph. For $N = 1$, we take the example of a rectangle building (48°07'54.7"N 1°40'42.2"W), where each node is either in line of sight with the base or is on the other side of the building. For $N = 2$, we consider a building with two rectangles (48°08'18.4"N 1°38'33.7"W). For $N = 3$, we choose a sophisticated building (48°07'49.3"N 1°37'43.6"W).

N =	$d = 50m$	$d = 35m$	$d = 20m$
1	0.02s ($n = 16$)	0.19s ($n = 32$)	4.8s ($n = 74$)
2	1.3s ($n = 30$)	12s ($n = 46$)	22s ($n = 110$)
3	23s ($n = 28$)	25s ($n = 42$)	32s ($n = 98$)

As N increases (meaning that the building are more and more sophisticated), the solving takes more time. Indeed, it is then harder to find a coverage, because we need to plan for the intermediate UAVs too. Notice the value of n has no clear impact on the time complexity, since the very complexity is due to the multi-hop aspect.

5 Future Work

We plan to generate benchmarks for the PDDL community, in the spirit of Section 4. We also plan to capture unpredictable environments in which the UAVs are progressing (weather, outdated building information, loss of UAVs, etc.); typically, the environment is modeled as an opponent player, as done in LTL synthesis [Pnueli and Rosner, 1989]. Indeed, we aim at generating plans that still achieve the goal, even with unpredictable events. In other words, the UAVs are connected enough to make sure that, at any point in time, if a mishap occurs, the connectivity of the UAVs is preserved. We also aim at extending our work to the imperfect information setting, where UAVs should make decisions according to their incomplete knowledge. We envision using recent tools developed in the model checking community such as MCMAS [Lomuscio *et al.*, 2017].

We also will include quantitative aspects, such as the duration of the mission, the battery levels, etc. It is fairly easy to add cost constraints, since it amounts to adding new conditions in the PDDL program. Yet, one may want to optimize a quantitative criteria to obtain a more fitting plan (shortest plans, least battery consumption, least number of drones, etc.).

References

[Francès *et al.*, 2017] Guillem Francès, Miquel Ramírez, Nir Lipovetzky, and Hector Geffner. Purely declarative action descriptions are overrated: Classical planning with simulators. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4294–4301, 2017.

[Giordano *et al.*, 2011] Paolo Robuffo Giordano, Antonio Franchi, Cristian Secchi, and Heinrich H. Bülthoff. Bilateral teleoperation of groups of uavs with decentralized connectivity maintenance. In *Robotics: Science and Systems VII, University of Southern California, Los Angeles, CA, USA, June 27-30, 2011*, 2011.

[Jørgensen and Lamarche, 2011] Carl-Johan Jørgensen and Fabrice Lamarche. From geometry to spatial reasoning : Automatic structuring of 3d virtual environments. In *Motion in Games - 4th International Conference, MIG 2011, Edinburgh, UK, November 13-15, 2011. Proceedings*, pages 353–364, 2011.

[Lomuscio *et al.*, 2017] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT*, 19(1):9–30, 2017.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl-the planning domain definition language. 1998.

[Pnueli and Rosner, 1989] Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, July 11-15, 1989, Proceedings*, pages 652–671, 1989.

[Teacy *et al.*, 2010] WT Luke Teacy, Jing Nie, Sally McClean, and Gerard Parr. Maintaining connectivity in uav swarm sensing. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1771–1776. IEEE, 2010.

[Yanmaz, 2012] Evsen Yanmaz. Connectivity versus area coverage in unmanned aerial vehicle networks. In *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*, pages 719–723, 2012.