

# Connected Multi-Agent Path Finding: Generation and Visualization

Arthur Queffelec<sup>1</sup>      Ocan Sankur<sup>2</sup>  
François Schwarzenruber<sup>1</sup>  
<sup>1</sup>Univ Rennes, CNRS, IRISA  
<sup>2</sup>Univ Rennes, Inria, CNRS, IRISA  
firstname.lastname@irisa.fr

May 27, 2021

## Abstract

We present a generic tool to visualize missions of the Connected Multi-Agent Path Finding (CMAPF) problem. This problem is a variant of MAPF which requires a group of agents to navigate from an initial configuration to a goal configuration while maintaining connection. The user can create an instance of CMAPF and can play the generated plan. Any algorithm for CMAPF can be plugged into the tool.

## 1 Introduction

In many applications (automated farming, search and rescue, etc.), unmanned autonomous vehicles (UAVs) have to stay connected at all time to share information. Introduced in [6] and extensively studied in [13, 3, 4], Connected Multi-Agent Path Finding (CMAPF) is the problem of generating such missions.

MAPF, from which CMAPF originated, asks for collision-free paths for a group of agents. See [14] for a comprehensive survey. However, from an algorithmic point of view, CMAPF is intrinsically harder than MAPF. It can be intuitively explained by the fact that a collision constraint is local (only concerns two agents) while the connectivity is global (it concerns the whole group of agents). While finding an arbitrary execution avoiding collisions is in PTIME [15], finding an arbitrary execution maintaining the connectivity is PSPACE-complete [13]. Optimisation problems for MAPF and CMAPF are NP-complete (see [4]). Algorithms have been developed to solve CMAPF [6, 13, 10].

As CMAPF has applications and is challenging, new algorithms will be developed. Thus, we need to visualize CMAPF instances and plans. Indeed, Visualization will help to understanding the generated plans, and thus to improve algorithms. It will also help to teach and explain these algorithms. A tool for the visualization of MAPF plans was developed [7]. However, to the best of our knowledge, no tool exists for

the visualization of CMAPF missions. The closest tool to our need was created in 2018 [1], in order to generate and visualize connected plan of coverage missions, where all locations of the graph must be visited by the agents. The latter work converts an instance of connected coverage planning into a classical planning instance. Coverage planning is close to CMAPF, as shown in [10]. The scalability was very limited and the collisions were not taken into account. Also the edition of maps was not possible.

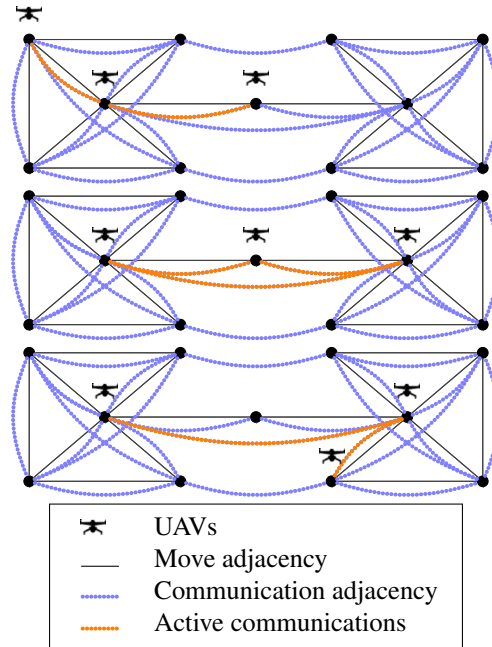


Figure 1: Example of a mission execution.

The goal of our tool<sup>1</sup> is to provide a platform for current and future algorithmic approach to visualize connected missions. It is worth noting that the connectivity constraints add a new layer of complexity to the missions compared to MAPF which render mission less comprehensible. The user can choose a 2D map on which to synthesize a connected cooperative plan. She can select the starting configuration of the agents and the goal configuration, the radius of communication and whether collisions are allowed or not. Then, the back-end algorithm generates a mission for the cooperation of agents.

The tool is provided with a simple algorithm to solve CMAPF. The structure of the algorithm is based on CBS [12] with positive constraints to deal with the particular challenges of CMAPF. Indeed, CBS deals with collision conflicts between two agents by enforcing one of the agents to move away from the location of the collision. In our algorithm, *connectivity conflicts* correspond to disconnected configurations, that is, configurations where the agents do not form a connected subgraph. Resolving such a conflict is harder: a connectivity conflict can involve multiple agents and a single

<sup>1</sup><https://github.com/francoisschwarzentruber/cmapf-gui>

disconnected agent may not be the source of the conflict.

Figure 1 shows a simple execution of 3-steps plan in a graph: solid lines represent elementary possible moves between two vertices; a dashed line between two vertices means that communication is possible between them. At the first step, agents are at the starting configuration. At the end, the agents reached their goal configuration. During the execution, agents stay connected (dashed lines forming a connected subgraph). Importantly, agents do not communicate while moving.

*Outline.* In Section 2, we explain the demonstration itself. Then, in Section 3 we describe the architecture of the tool. In Section 4, we describe our algorithm. We conclude in Section 5.

## 2 Demonstration Outline

### 2.1 Choose an instance

The software starts by showing a map. The user can select another map. She can also modify the radius of communication, that is the maximum distance between two locations in which two agents could communicate. She can choose to add new agents. Then, the user selects the starting locations of all the agents and the goal locations to reach. As seen in Figure 2a, the user places the circles for the starting positions of the agents and the flags for the ending positions.

### 2.2 Visualize generated plans

Once the mission is specified, the user can ask for the mission to be computed. The algorithm computes the solution (or lack of solution) and informs the user. If a plan exists, the user can scroll through the plan and see every step. In particular, the user can go back and forth in the plan.

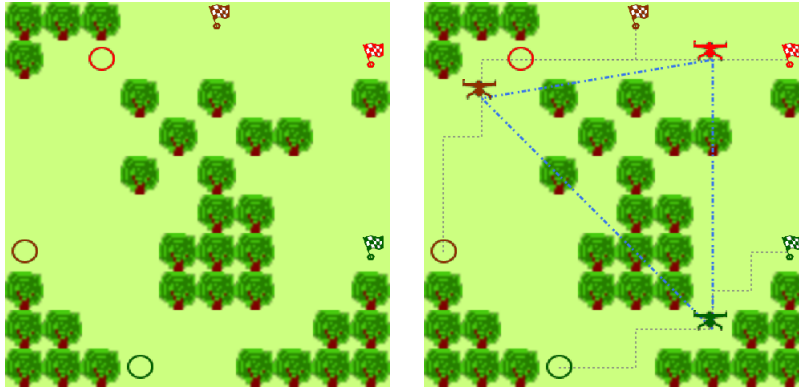
In Figure 2b, the path of each agent is displayed with a gray dashed line on the ground and the connectivity between the agent at each step is shown by a blue link between the inter-connected agents.

### 2.3 Create new maps

We wanted users to be able to create easily their own maps. That is why we decided to represent maps as 2D black and white bitmaps (see Figure 3). That representation makes the creation and edition of maps easy by means of any drawing software. Black pixels are the obstacles while the white one are the nodes of the graph. The movement is allowed between two adjacent white pixels. In the current version of the tool, the user specifies a radius of communication.

## 3 Architecture of the Tool

In this section, we describe the architecture of our tool, as shown in Figure 4. Importantly, the visualizer is independent from the underlying algorithm that is used. So we



(a) Selection of an instance. The user can drag and drop the starting and target locations of the agents. (b) Execution of a plan. Paths of agents are shown, as well as the connectivity between agents.

Figure 2: Screenshots of the tool.



Figure 3: A map is represented by a black and white bitmap.

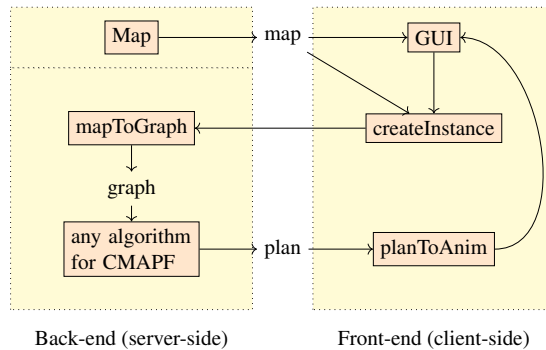


Figure 4: Architecture of the tool

can easily plug any other algorithm.

**Front-end** The front-end of the tool displays the map (in which trees are obstacles) and allows the user to construct an instance. When the user decides to compute an execution, it generates an instance corresponding and send it to the back-end for processing. The plan of the mission is then transformed into an animation that the user can play. The front-end is written in TypeScript.

**Back-end** The back-end of the tool transforms the map to a pair of two graphs (movement and communication). The graphs are described in the GraphML format (making it easy to load the graphs e.g. in C++ via the library Boost or rapidxml). Then, it contains an algorithm that takes the graphs and the start and goal configurations as an input. Finally, the algorithm generates a plan or inform the lack of feasible plan. The returned plan is a sequence of actions such that the connectivity is maintained.

## 4 Algorithm

Our visualization tool is supplied with an algorithm for solving CMAPF. We provide a lightweight implementation in C++ (in comparison, the implementation given in [13] requires Robot Operating System). Our algorithm is inspired by the concept of the Conflict-Based Search (CBS) algorithm [12] which solves MAPF by finding an optimal collision-free execution (without any connectivity constraints). CBS searches for an execution using a *decoupled* approach, computing the path of each agent independently and repairing any issue. This approach does not suffer directly from the state explosion problem. Indeed, CBS is intuitively exponential in the number of collision and not in the number of agents.

As in CBS, our algorithm Connected CBS (CCBS) is composed of two levels: the high-level builds a constraint tree to search for a set of constraints; the low-level computes path for single agents satisfying a given set of constraints. A *constraint* can

either be positive or negative and applies to an agent, a time-step and a location. Notice that [12] only uses negative constraints, while we use both negative and positive ones.

**High-Level** The high-level of CCBS builds a constraint tree in order to find a set of constraints which allows the low-level to generate a valid execution. CCBS starts with a tree of a single constraint node containing no constraint. The low-level is used to obtain an execution. If the execution is valid, we return it. Otherwise, we pick a conflict and *split* it the following way.

- Collision Conflict: pick a pair of agents in collision, generate two negative constraints, each enforcing one of the agent away.
- Disconnection Conflict: pick an agent, for each possible position, generate a positive constraint enforcing the agent at this location.

This process repairs all conflicts in a best-first search manner.

**Low-Level** We use the algorithm described in [9] to compute the constrained shortest paths for individual agents with positive and negative constraint.

As CBS, in the case of a not satisfiable instance, our algorithm may not terminate.

## 5 Conclusion

The tool presented in this paper offers a framework for researchers to connect their algorithms for CMAPF. For the moment, the communication type is limited to radius; we aim at adding other communication types (e.g. line-of-sight). We plan to enrich the graphical user interface so that the user may directly choose other algorithms (in particular SB, DFS from [13]). A variant of CMAPF includes a base, that is, a specific agent that can not move (see [1, 4, 10]). We also would like to enrich the graphical user interface for capturing the notion of base, or more generally of agents that are not allowed to move. In this line, we may also imagine agents will different abilities and take the battery energy levels in the planning problem.

As said in the introduction CMAPF is a challenging problem. We aim at adapting optimization coming from CBS for MAPF. For instance, for CBS, a popular and efficient optimization considers *cardinal* conflicts [2], which are conflicts that can only be solved by increasing the lengths of the paths of all involved agents. This concept led to the integration of powerful heuristics in CBS [5, 8]. The adaptation of similar concepts for connectivity is however nontrivial.

The CMAPF problem was extended to partially known environments in [11]. In this setting the agents do not know exactly the environment in which it is moving and have to discover it during their mission. Thus, when agents connect together they are allowed to share their information. We also would like our tool to support partially known environment. The user would be capable of observing the point of view of each agent, its current knowledge and the knowledge gained through other agents.

## References

- [1] F. Bodin, T. Charrier, A. Queffelec, and F. Schwarzentruber. Generating plans for cooperative connected uavs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5811–5813. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
- [2] E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and S. E. Shimony. ICBS: improved conflict-based search algorithm for multi-agent pathfinding. In *IJCAI 2015*, pages 740–746, 2015.
- [3] T. Charrier, A. Queffelec, O. Sankur, and F. Schwarzentruber. Reachability and coverage planning for connected agents. In *IJCAI 2019*, pages 144–150, 2019.
- [4] T. Charrier, A. Queffelec, O. Sankur, and F. Schwarzentruber. Complexity of planning for connected agents. *Auton. Agents Multi Agent Syst.*, 34(2):44, 2020.
- [5] A. Felner, J. Li, E. Boyarski, H. Ma, L. Cohen, T. K. S. Kumar, and S. Koenig. Adding heuristics to conflict-based search for multi-agent path finding. In *ICAPS 2018*, pages 83–87, 2018.
- [6] G. A. Hollinger and S. Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics.*, pages 967–973, Aug 2012.
- [7] P. Koupy. Visualization of problems of motion on a graph. <http://koupy.net/graphrec.php>, 2010.
- [8] J. Li, A. Felner, E. Boyarski, H. Ma, and S. Koenig. Improved heuristics for multi-agent path finding with conflict-based search. pages 442–449, 08 2019.
- [9] J. Li, D. Harabor, P. J. Stuckey, A. Felner, H. Ma, and S. Keonig. Disjoint splitting for conflict-based search for multi-agent path finding. In *ICAPS 2019*, pages 279–283, 2019.
- [10] A. Queffelec, O. Sankur, and F. Schwarzentruber. Conflict-based search for connected multi-agent path finding. *CoRR*, abs/2006.03280, 2020.
- [11] A. Queffelec, O. Sankur, and F. Schwarzentruber. Planning for connected agents in a partially known environment. In *Advances in Artificial Intelligence - 34th Canadian Conference on Artificial Intelligence, Canadian AI 2021, Kingston, ON, Canada, May 25-28, 2021, Proceedings*, 2021.
- [12] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.*, 219:40–66, 2015.
- [13] D. Tateo, J. Banfi, A. Riva, F. Amigoni, and A. Bonarini. Multiagent connected path planning: PSPACE-completeness and how to deal with it. In *AAAI 20018.*, pages 4735–4742, 2018.

- [14] J. Yu and S. M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.
- [15] J. Yu and D. Rus. Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In H. L. Akin, N. M. Amato, V. Isler, and A. F. van der Stappen, editors, *Algorithmic Foundations of Robotics XI - Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics, WAFR 2014, 3-5 August 2014, Boğaziçi University, İstanbul, Turkey*, volume 107 of *Springer Tracts in Advanced Robotics*, pages 729–746. Springer, 2014.