Plan merging

Aurélien SIMON

July 21, 2022

Abstract

We are talking about plan merging here

1 Defining what can be de output of IPF

We are working on a grid represented by a graph. Non-anonymous MAPF. Makespan, working with edge conflict and edge conflict only

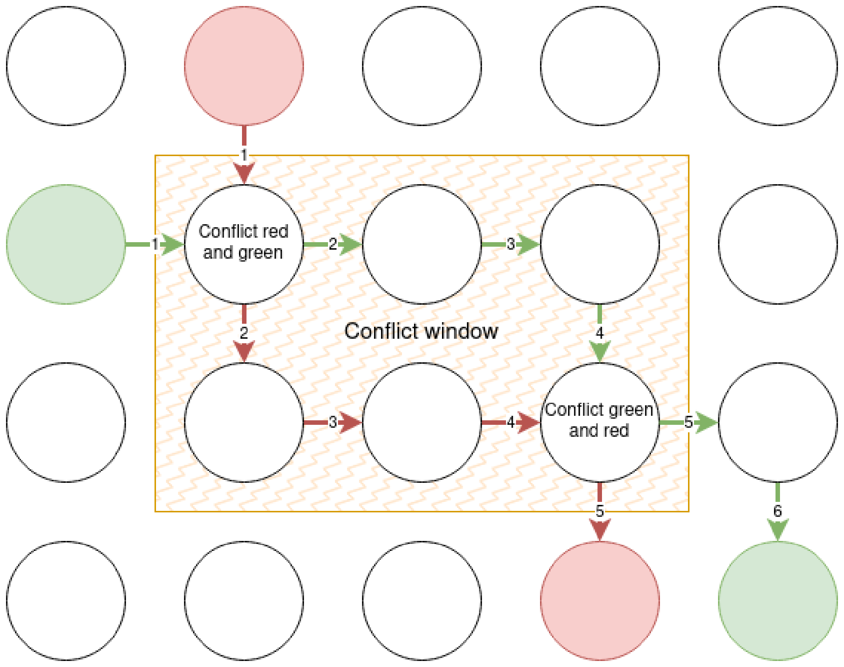
1.1 Definition

Definition 1.1 (path)**.** A path is a sequence of nodes where the initial node cannot be the same as the last node in the sequence. For each successive node in the sequence, an edge betwenn the two node must exist.

Definition 1.2 (Shortest path)**.** Given a initial node and a final node for the sequence, a shortest path of length n exist if no other path of length $n' < n$ exist for the same intial and final node.

Definition 1.3 (Conflict window)**.** A conflict window[13] is a rectangular[1] area of the graph that contains all the conflict (path or plan).

Figure 1: Conflict window

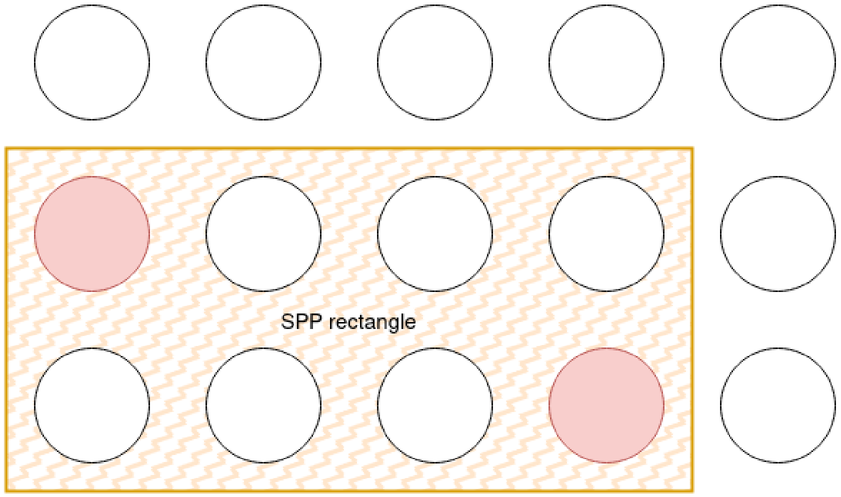


Definition 1.4 (Shorter shortest path)**.** A shortest shortest path SSP is a path contained[2] in a rectangle where the goal and the initial position define its diagonal. Furthermore, this path must use a most 2 different direction.

[1]it can actually be any geometrical shape, circle might be interestiong as well
[2]means that for every step of the plan, each position is in the rectangle

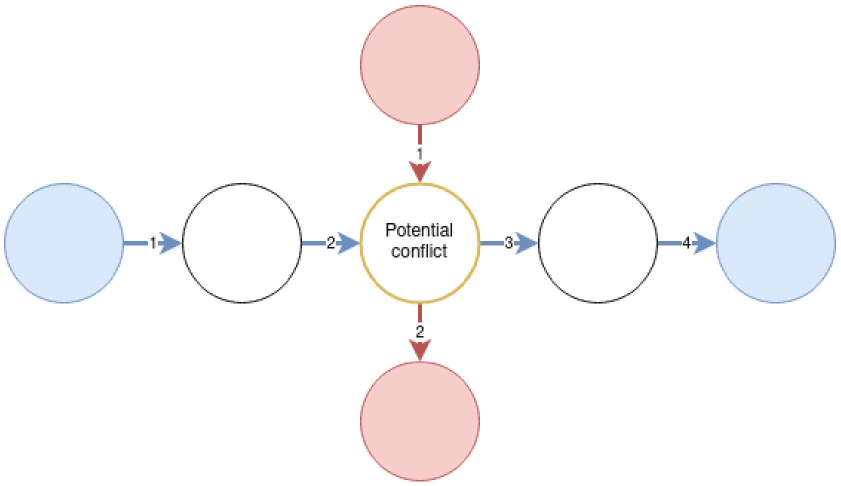
Definition 1.5 (Longer shortest path)**.** A longer shortest path LSP is a path that have a least one node and its sequence that is not contained in the SPP rectangle.

Figure 2: Shorter shortest path rectangle



Definition 1.6 (Potential conflict or path conflict)**.** Potential conflict or path conflict happened when a node-conflict exist without considering time step; considering two agents a and a' and their plan π_a and $\pi_{a'}$, a potential conflict exist at node n iff $n \in \pi_a$ and $n \in \pi_{a'}$.

Figure 3: Potential conflict



Definition 1.7 (Diversity)**.**

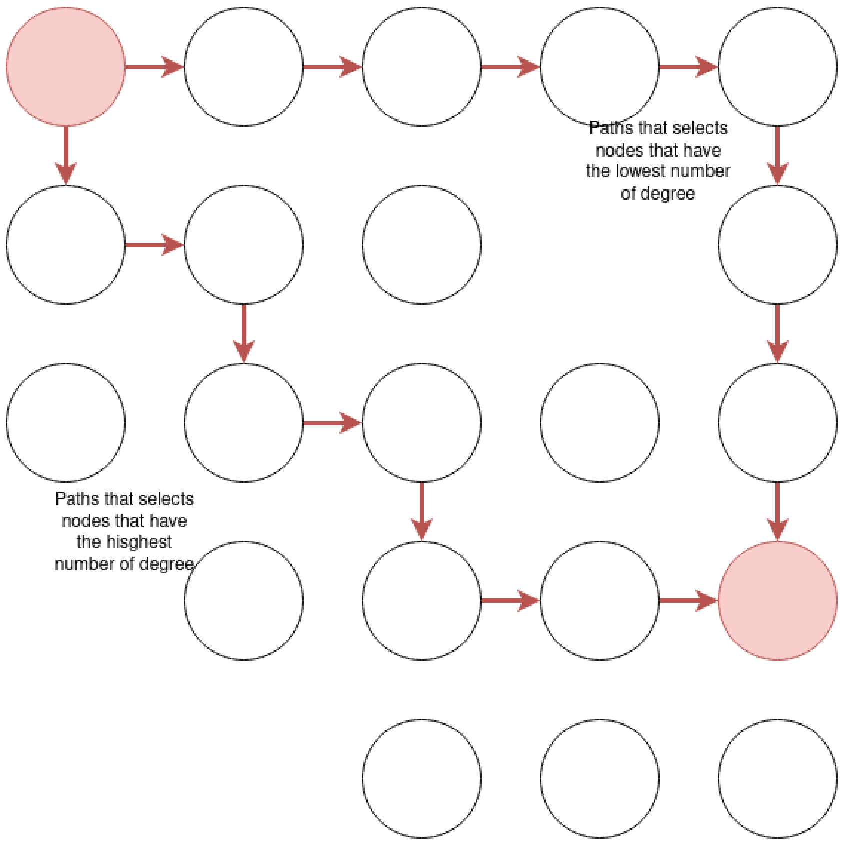
Definition 1.8 (Distance (between paths))**.**

1.2 One path: criteria

We can first consider the case where IPF gives as output (input of plan merging) only 1 shortest (we can consider also paths that are longer than the shortest path) path in SP for each agent. Thus we can have:

- Random shortest path
- Shortest path that contains n bending.
- path that goes through nodes that have (recursively?) the highest degree. (Goes into the middle)
- path that goes through nodes that have the lowest degree. (Goes next to wall)

Figure 4: Example of paths based on the degree of nodes


Paths that selects nodes that have the lowest number of degree
Paths that selects nodes that have the hisghest number of degree

- Path that contains n bendings.

Note that for each selection of paths, we can always consider the opposite approach, for instance minimizing instead of maximizing.

1.3 Selection of one path considering other agents

This section describe paths that are dependent of other agents paths.

- path that minimize path conflicts
- path that minimize plan conflicts
- path that minimize conflict window size for path conflict or plan conflict (Building More Compact Models [6])
- path that maximize diversity with other agents
- path that maximize distance with other agents
- path that maximize following conflict
- path that maximize different objectives[12]
- path that contains no path conflict with the highest number of agent

1.4 A set of paths

Then we can consider that IPF gives us n paths as output:

- k random shortest paths. k can be an arbitrary number, a function that gives number of paths to pick[3, 9].
- all shortest paths
- 3 distant paths[9].
- k random shortest paths + k paths that have a length of $|SP| + 2n_{\in \mathbb{N}}$
- n paths that are in SSP and n' paths that are in LSP

Overall we can use a combinaison of every single path strategies.

2 Merging

2.1 Systematic approach

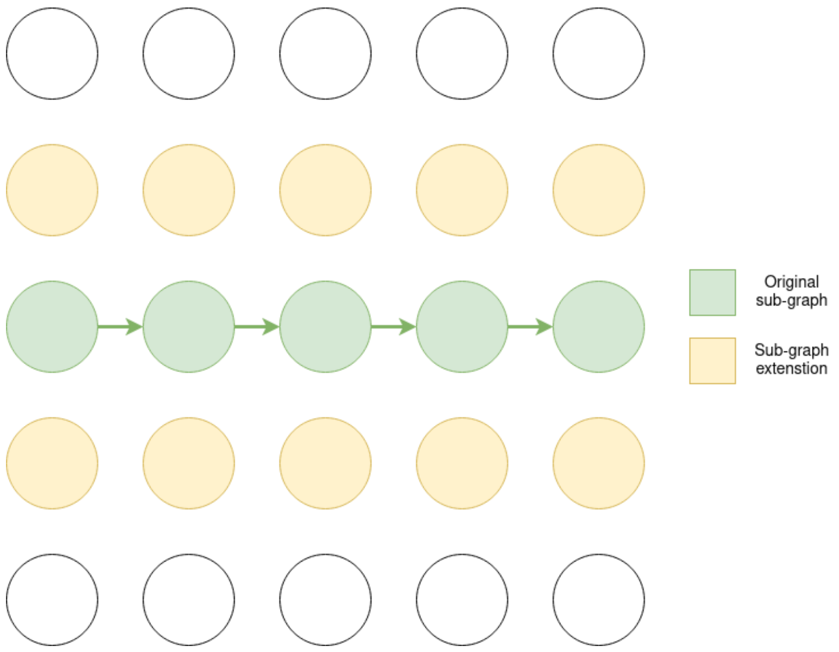
In order to merge paths to obtain a correct solution, we can use the approach descibed in the paper [9] as a systematic approach. For each agents and their paths, we create a disjoint graph that is a sub graph based on the paths of the agent. We can now consider two different possibilities; 1. Input paths can create a collision free solution and the output would be the ouput of IPF. 2. Their is no conflict free solution. In this case we can extend the graph (nodes or egdes) and give more times for the agents to reach their goals (complete and optimal).

2.2 Extending the sub-graphs

2.2.1 Corridor

- Summary: considering a sub-graph based on the path of a agent. The corridor heuristics will extend the sub-graph by adding every neighboor nodes of the sub-graph. Furthermore, the extension can have different levels.

Figure 5: Example of corridor

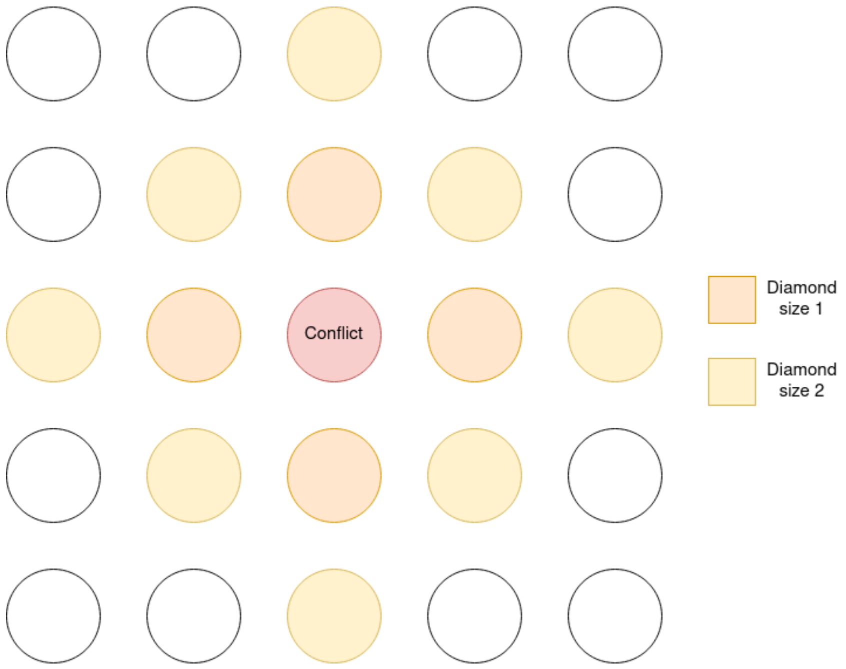


- Require: Can work with multiple paths, but having distant path would be better since the extension won't overlap. We can also considering selecting paths that have less variety between each agents so the corridor would become sort of a highway.
- Variant

2.2.2 Diamond extention

- Summary: From the paths of the agents, for each node where a conflict is identified, we extend the sub graph of the conflicting agents around the node of the conflicting agent
- Require: Can work with on or multiple path, however, adding more path might create too much extension ending up covering the whole graph

Figure 6: Example of diamond of size 1 and 2



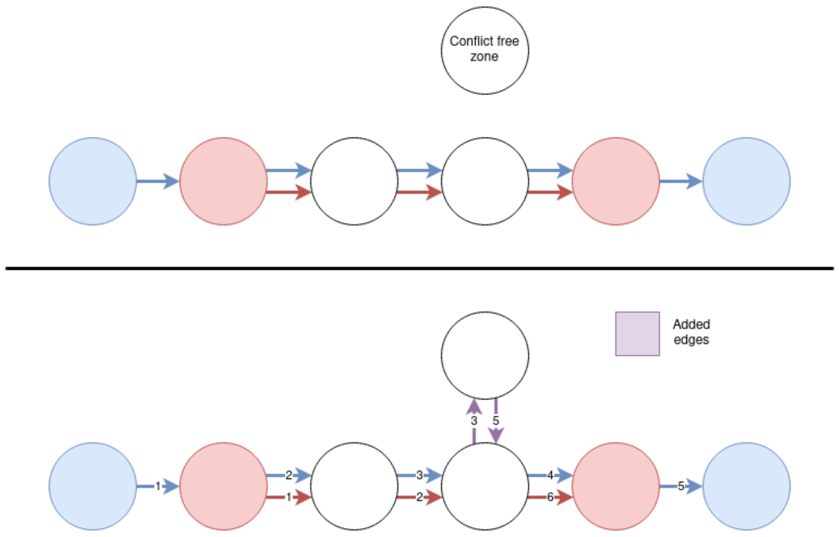
2.2.3 Plan repair[7] via neighboor search

- Summary: starting from an inital path, the idea would be to extend this inital path by looking into the neighboor solution starting from a conflict point. Maybe other approach are part of neightboor search[8].

2.2.4 Blocking agents solving

Definition 2.1 (Blocking agent)**.** A agent is called a blocking agent if its goal or its intial position is on the path of anoter agent. It is also considered as fully blocking if both inital and goal position is on the way of a single other agent. Assumption: a full-blocking agent needs to extend his subgraph innn order to find a valid solution.

Figure 7: Example of blocking agent



- Summary: consiering the definition above, the idea would be to force blocking agents to reach a conflict free zone and let the non-blocking agents reach their goal first. As shown in **??**.
- Require: can work with any type or number of path, however, using $|SP|+2n_{\in\mathbb{N}}$ paths might solve the cconflict by default without having to look for the closest conflict-free zone
- Variant: Conlict-free zone can evolve over time in order to have more flexibility (time interval)

2.3 Guiding the agents

2.3.1 Heatmap

- Summary: The idea is to create for each nodes of the graph, an indicator that would describe how much the node "used" b y the agents. We can imagine something like, for each node, we have the number of timestep where an agent is at the node divided by the makespan. The idea would then be to
  - guide through probability the agents
  - guide the graph extension by choosing the nodes that have a lower or higher "usage"
- Require: at least one path for each agents, the more paths we have the more precise the heat map can be.
- Variants:

- Take into consideration the robustness [1, 2]
- make the heatmap evolve over timestep

2.3.2 From the bachelor thesis

- Deathlaser: the idea is to keep the agents at a closer position from their goal by pruning their subgraph around their prevísous position, probably require only 1 paths
- Homesick: the idea is to force the agent to come back to their plan after t time not being on their origianl plan. Might require a graph extention beforehand
- Checkpoints (or Divide and Conquer [6]): the idea is to add n "checkpoint" nodes on the original path where their final paths should contains these nodes. Work only with one path

2.4 Others

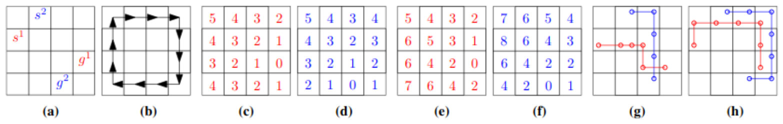
2.4.1 Time expanded graphs

Need to explore more about this; how it can help solving the problem, what are the kind of path that would be interesting as input, what merging strategies can be applied..

2.4.2 Highway

- Summary: Highway[4, 5] heuristic aims to change the graph by remove edges resulting with part of the graph that become oriented.
- Require: using following conflict paths might create better highway especially on self generated[4] ones.

Figure 8: Example of highway (figure 2 of paper[5])

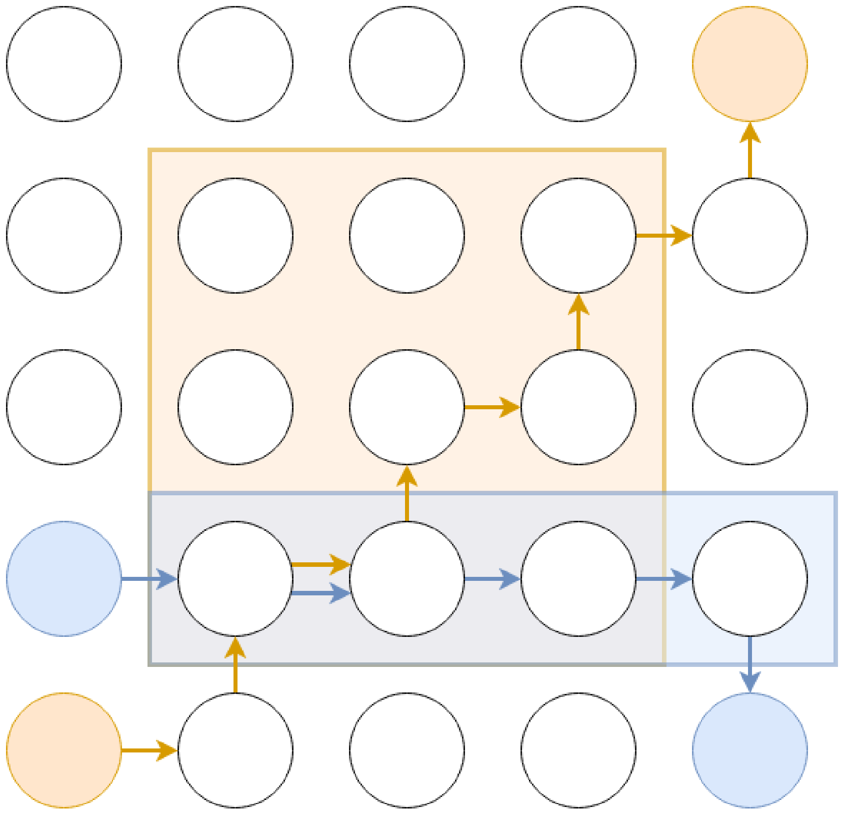


2.4.3 Pruning by pulling goal

- Summary: Considering a agent and its path, each step both the goal and the agent are moving toward each other until a path or plan conflict occurs. Occe a conflict is found, we use the position of the goal and the position of the agent to a rectangle (being the diagonal).

- Require: feels like it could only work for one path at the time, otherwise it might take too much computation time.

Figure 9: Example of goal pulling



2.4.4 Loop Highway 14-Loyd-Puzzle

- Idea: considering a path that goes through every starting position and goal position of every agents, loop, and do not cross; we can meybe assume that if the loop has two free neighboor, their is a solution. If we consider that agent disapear on goal, it is an even better solution. If we can create the loop talked about above we are in a situation in worst case of the 14-Loyd-puzzle that I assume is solvable whatever the configuration.

2.4.5 Waiting only

- Summary: the agents are only allowed to wait, no sub-graph extention is allowed.

- Require: In order to raise the chances of finding a solution, we need a higher number of diverse path, however, som problems can't be solved through waiting, and some problem could be solved by waiing a huge amount of time.

- Variant:
  - Fixing waiting position with time interval [10, 11]

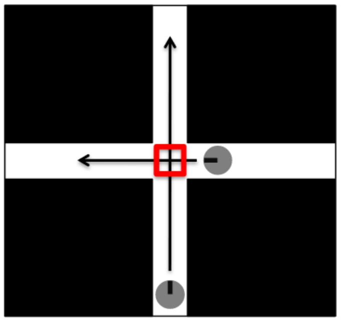


Fig. 2. An environment with dynamic obstacles and a highlighted configuration

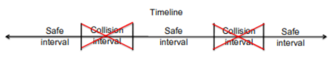


Fig. 3. A timeline for the highlighted configuration in Figure 2

  - Fixing waiting position based on conflict free zone

References

[1] D. Atzmon et al. "Probabilistic Robust Multi-Agent Path Finding". In: *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS'20)*. Ed. by J. Beck et al. AAAI Press, 2020, pp. 29–37.

[2] D. Atzmon et al. "Robust Multi-Agent Path Finding and Executing". In: *Journal of Artificial Intelligence Research* 67 (2020), pp. 549–579.

[3] Martim Brandão and Yonathan Setiawan. "'Why Not This MAPF Plan Instead?'Contrastive Map-based Explanations for Optimal MAPF". In: *ICAPS 2022 Workshop on Explainable AI Planning*. 2022.

[4] L. Cohen et al. "Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding". In: *Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. Ed. by R. Kambhampati. IJCAI/AAAI Press, 2016, pp. 3067–3074.

[5] Liron Cohen and Sven Koenig. "Bounded Suboptimal Multi-Agent Path Finding Using Highways." In: *IJCAI*. 2016, pp. 3978–3979.

[6] J. and S. LaValle. "Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics". In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1163–1177.

[7] Antonın Komenda et al. "How to repair multi-agent plans: Experimental approach". In: *Proceedings of Distributed and Multi-agent Planning (DMAP) Workshop of 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*. 2013.

[8] Jiaoyang Li et al. "MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022.

[9] "Multi-agent Pathfinding on Large Maps Using Graph Pruning: This Way or That Way?" In: (2022).

[10] Venkatraman Narayanan, Mike Phillips, and Maxim Likhachev. "Anytime safe interval path planning for dynamic environments". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 4708–4715.

[11] Mike Phillips and Maxim Likhachev. "Sipp: Safe interval path planning for dynamic environments". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 5628–5635.

[12] Zhongqiang Ren, Sivakumar Rathinam, and Howie Choset. "Subdimensional Expansion for Multi-Objective Multi-Agent Path Finding". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7153–7160. DOI: `10.1109/LRA.2021.3096744`.

[13] Kyle Vedder and Joydeep Biswas. "X*: Anytime multi-agent path finding for sparse domains using window-based iterative repairs". In: *Artificial Intelligence* 291 (2021), p. 103417.