Scalable Task and Motion Planning for Multi-Robot Systems in Obstacle-Rich Environments

Doctoral Consortium

Wolfgang Hönig
University of Southern California
whoenig@usc.edu

ABSTRACT

Motion planning problems have been studied in both the artificial intelligence (AI) and robotics communities. AI solvers can compute plans for hundreds of simple agents in minutes with suboptimality guarantees, while robotics solutions typically include richer kinodynamic models during planning, but are very slow when many robots and obstacles are taken into account.

We combine the advantages of the two methods by using a two-step approach. First, we use and extend AI solvers for a simplified coordination problem. The output is a discrete plan that cannot be executed on real robots. Second, we apply a computationally efficient post-processing step that creates a continuous plan, taking kinodynamic constraints into account. We show examples for ground robots in a warehouse domain and quadrotors that are tasked with formation change.

ACM Reference Format:
Wolfgang Hönig. 2018. Scalable Task and Motion Planning for Multi-Robot Systems in Obstacle-Rich Environments. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

Coordinating many collaborative robots is very useful for search-and-rescue, mining, entertainment, and warehouse automation. In many of those scenarios, robots have to move in tight spaces filled with obstacles to reach their objectives. In artificial intelligence, path-finding planners find solutions quickly but make simplifying assumptions: both time and space are discretized and the agents are assumed to be point robots that can move in perfect synchronization on a graph. The robotics community has developed solutions that use detailed models of the robots including kinodynamic constraints. Combined with robust controllers, this ensures that the computed plans can likely be executed on real robots in practice. However, such planning is computationally expensive and often only works with a team of a few robots.

In this work, we combine the advantages of the two approaches. We develop a scalable algorithm that comes with theoretical guarantees and apply it to real robot systems. Our approach is based on planners from the AI community (and novel extensions thereof) and uses post-processing steps to allow safe execution on real robots. We leverage the fact that AI solvers solve the coordination problem on an abstract level, creating a partial order between the robots.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

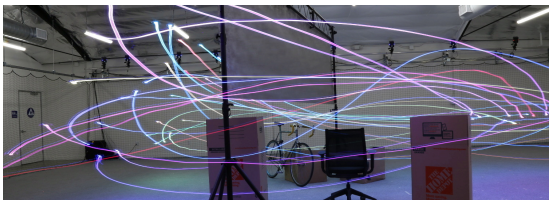

Figure 1: Long exposure of 32 Crazyflie nano-quadrotors flying through an obstacle-rich environment.

We can refine such an abstract solution to include robot-specific properties. The post-processing step depends on the type of robots, and we provide examples for ground robots and aerial vehicles.

2 AGENTS

We first introduce *Multi-Agent Path-Finding* (MAPF) and later extend it to account for the physical extent of robots. The environment is represented in a search graph $\mathcal{G}_E = (\mathcal{V}_E, \mathcal{E}_E)$ with unit-length edges. The set of agents is $\{1, \dots, N\}$. For agent j let u_t^j be the vertex that is occupied at timestep t, $s^j = u_0^j$ be the given start vertex, and g^j the goal vertex. The discrete schedule for agent j is $p^j = [u_0^j, \dots, u_{T^j}^j, u_{T^j+1}^j, \dots]$. The goal for a MAPF solver is to find a discrete schedule for each agent, such that: 1. Every action is either a move action along an edge or a wait action; 2. Agent j remains at a vertex g^j after timestep T^j; 3. Two agents do not occupy the same vertex at the same timestep; and 4. Two agents do not traverse the same edge in opposite directions at the same timestep. An optimal solution might minimize the *makespan* ($\max_j T^j$) or the *total time* ($\sum_j T^j$). Solving MAPF optimally is NP-hard, but w-suboptimal solvers exist that can solve problem instances with hundreds of agents quickly in practice.

The existing MAPF formulations assume that two agents can safely traverse two different edges simultaneously. This is not true in general roadmaps that might be created with algorithms such as PRM*, because edges might be arbitrarily close to each other. We introduce *MAPF with generalized conflicts* (MAPF/C). We operate on the search graph $\mathcal{G}_E$ with additional conflict sets for generalized vertex-vertex (*conVV*), edge-edge (*conEE*), and edge-vertex (*conEV*) conflicts. A discrete schedule has to fulfill the following additional properties: 5. Agents obey inter-agent constraints when stationary (*conVV*); 6. Agents obey inter-agent constraints while traversing an edge (*conEE*); and 7. Agents obey inter-agent constraints between stationary and traversing agents (*conEV*).

We develop solvers for MAPF/C based on solvers for MAPF that can find solutions for the labeled and unlabeled variants [2].

3 GROUND ROBOTS

We now consider ground robots, such as differential drive robots, that operate in warehouse environments. We demonstrate our approach with a simple example of two robots in a narrow corridor of a grid-world, see Fig. 2 (top). The two robots have different maximum speed limits and edge (C, D) has a minimum speed limit. Robot 1 must pass Robot 2 to reach its goal location, which requires Robot 2 to move into an alcove temporarily, no matter what the speed limits are. We can thus use a discrete solver from AI to discover such critical intermediate configurations and then use our post-processing step MAPF-POST to create a continuous schedule.

MAPF-POST uses a *Temporal Plan Graph* (TPG) as data structure. A TPG is a directed acyclic graph $\mathcal{G}_{TPG} = (\mathcal{V}_{TPG}, \mathcal{E}_{TPG})$. Each vertex v represents an event, which corresponds to a robot entering a location. Each edge (u, v) is a temporal precedence between events u and v indicating that event u must be scheduled no later than event v. We can construct a TPG given a discrete schedule in polynomial time. We use two different kinds of temporal precedences: Type 1 edges enforce that a robot enters locations in the order given by its discrete schedule (black edges in Fig. 2) and Type 2 edges enforce the order in which two different robots enter the same location in the discrete schedules (colored edges in Fig. 2).

The basic TPG does not provide any safety distance between robots. We can add additional vertices (called safety markers) to the TPG to provide a guaranteed safety distance between robots. The safety markers correspond to new locations and can be added given a user-specified safety distance δ. The augmented TPG can be constructed in polynomial time and requires more vertices and edges the smaller δ gets. The safety markers are the unlabeled vertices in Fig. 2 and were created using $\delta = 0.25\,\text{m}$ (all edges in the environment are 1 m long).

We can now transform the TPG into a *Simple Temporal Network* (STN), where each edge $e = (u, v)$ is annotated with bounds $[LB(e), UB(e)]$ indicating that event u must be scheduled between $LB(e)$ and $UB(e)$ time units before event v. This annotation can directly take minimum and maximum speed limits of the robots and edges into account. A solution that minimizes makespan to the STN can be computed in polynomial time and assigns a continuous arrival time to each location. The STN can be re-solved quickly online to account for execution deviations.

We showed that MAPF-POST guarantees the user-specified safety distance and provided experimental results with a group of robots [1].

4 AERIAL VEHICLES

We now consider the formation change problem for a team of quadrotors in obstacle-rich environments. While it is possible to use MAPF-POST for such a problem, the quadrotors would need to stop at each vertex and could not fly in close vertical proximity because MAPF-POST ignores the so-called downwash effect.

Quadrotors generate a large, fast-moving volume of air underneath their rotors called *downwash*. The downwash force is large enough to cause a catastrophic loss of stability when one rotorcraft flies underneath another. We model downwash constraints as inter-robot collision constraints by treating each robot as an axis-aligned ellipsoid. Our approach first discretizes the problem and constructs a MAPF/C instance. The discrete schedule is used as input to a trajectory optimization problem that iteratively creates smooth trajectories.

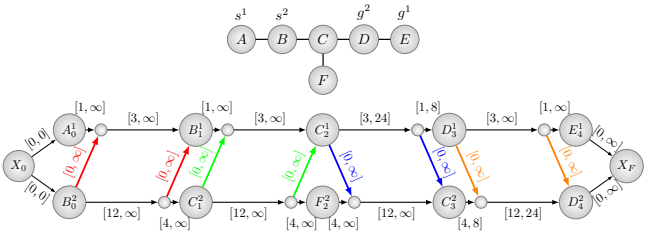

Figure 2: MAPF-POST Example.

The discrete portion consists of three parts. First, we generate a roadmap given a map of the environment, the shape of the robots, and a set of start and goal locations, using the SPARS algorithm. Second, we annotate the roadmap with vertex-vertex (*conVV*) conflicts (pairwise collision checking between ellipsoids placed at vertices), edge-edge (*conEE*) conflicts (pairwise collision checking between swept ellipsoids along edges), and edge-vertex (*conEV*) conflicts (collision checking between swept ellipsoid along edge and ellipsoid at vertex). Third, we use the annotated roadmap as input for MAPF/C and find a collision-free discrete schedule for all robots.

The trajectory optimization first finds *safe corridors* within the free space for each robot. For each timestep k and robot i we find the convex polyhedron $\mathcal{P}_k^i$ by intersecting the half-spaces that separate the current robot from all other robots and the half spaces that separate the current robot from all obstacles. The union $\cup_k \mathcal{P}_k^i$ defines a safe corridor for robot i. We base the trajectories on Bézier curves, which are guaranteed to lie in the convex hull of all control points. During optimization, we find a Bézier curve per timestep k and robot i and constrain its control points to be within $\mathcal{P}_k^i$, guaranteeing that the resulting curve will be within $\mathcal{P}_k^i$ also. The optimization is independent of the other robots and can be executed in parallel. We can iteratively repeat the optimization to improve the quality of the trajectories. In the last step, we stretch the trajectories such that all dynamic limits (e.g., maximum acceleration) are fulfilled.

We provide simulations with up to 200 quadrotors and execute the trajectories on 32 real quadrotors [2], see Fig. 1.

5 CONCLUSIONS

We have developed a method that can compute trajectories for hundreds of robots in obstacle-rich environments within minutes. Our approach combines ideas from artificial intelligence and robotics. We use AI planners with simplistic agent models, perform a computationally efficient post-processing step, and execute the resulting plan on physical robots with safety guarantees. For ground robots, our post-processing step is MAPF-POST, an algorithm based on simple temporal networks that can take into account simple kinematic constraints (such as speed limits) and a safety distance. The post-processing step for UAVs is based on trajectory optimization for each robot within a safe corridor. We demonstrate the approach in simulation and on up to 8 ground robots and 32 UAVs.

In the future we plan to investigate robust plan execution for persistent operation even when the environment or tasks change.

REFERENCES

[1] Wolfgang Hönig, T. K. Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. 2018. Path Finding for Multi-Robot Systems with Kinematic Constraints. *Journal of Artificial Intelligence Research (JAIR)* (2018). Accepted.

[2] Wolfgang Hönig, James A. Preiss, T. K. Satish Kumar, Gaurav S. Sukhatme, and Nora Ayanian. 2018. Trajectory Planning for Quadrotor Swarms. *IEEE Transactions on Robotics, Special Issue on Aerial Swarm Robotics* (2018). Accepted.