Multi-Agent Task and Motion Planning: An Optimization based Approach

Camille Phiquepal **Marc Toussaint**
Machine Learning & Robotic Lab, University of Stuttgart
{firstname.surname}@ipvs.uni-stuttgart.de

***Abstract*— We propose a new algorithm for Multi Agent Task and Motion Planning (TAMP). Our approach builds on the Logic-Geometric Programming framework (LGP) presented in prior work [1, 2]. The presented algorithm plans policies that react to the actions of the other agents, both on the symbolic and the motion level. To this end, we optimize *trajectory trees* that describe the branchings of optimal motions depending on the other agent actions. The algorithm works in two stages: First, the symbolic policy is optimized using approximate path costs estimated from independent optimization of trajectory pieces. Second, we fix the best symbolic policy and optimize a joint trajectory tree.**

I. Introduction

Our research interests lie in Task and Motion planning in the presence of *uncertainty*. Our past research focused on the uncertainty induced by partial observability [3]. This abstract presents an extension to the case of uncertainty induced by the behavior of other agents.

II. Problem Formulation

A. Decision tree

To define a problem we first define a multi-agent decision process. Second, for each action, we define cost and constraint functions that define the continuous trajectory problems associated with actions. The problem can then be defined in terms of a decision tree alternating two kinds of nodes: ego-nodes (representing a decision of the controlled agent), and nodes representing the possible decisions of the other agents. In the case of two agents acting adversarially, this boils down to a min-max-tree. An optimal policy is then comprised of a reactive symbolic policy π^* that transitions the tree depending on the other agents actions, and an optimal *trajectory tree* ψ^* which, smoothly switches into different motion options.

1) Example of decision tree: We consider a car behind a truck with a car coming in front, see Fig. 1. The ego-car wishes to overtake. At the first time step, the agent can either initiate the overtaking (change lane), or stay in line. After starting to overtake, the ego-car can accelerate to overtake, or move back behind the truck. In the meantime, the behavior of the car coming in front is uncertain, and either slows-down, accelerates, or continues at its current speed. Fig. 2 is the decision tree.

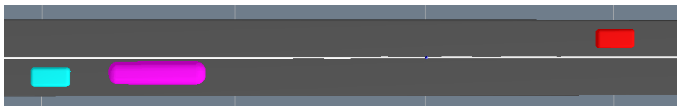
Fig. 1: The ego vehicle (cyan) wants to overtake although the behavior of the red car coming in front is uncertain.

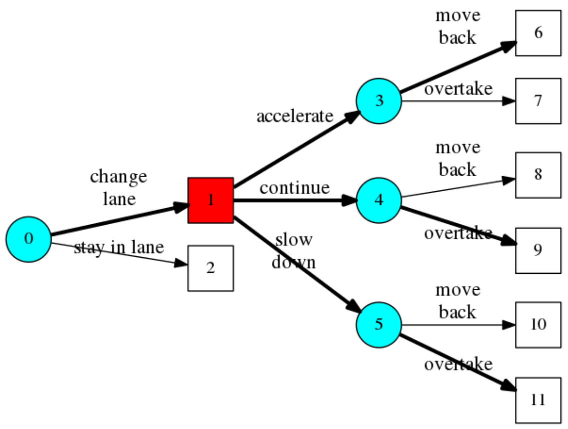

Fig. 2: Decision tree for overtaking. The cyan nodes represent decisions of the ego-car, red nodes are for the car in front. Thick edges represent a possible policy

B. Trajectory Tree

Let x be a trajectory, taking action a over $[t_k, t_{k+1}]$ implies costs

$$c(a,x) = \int_{t_k}^{t_{k+1}} f_a(x(t), \dot{x}(t), \ddot{x}(t))dt \tag{1}$$

$$s.t \quad g_a(x(t), \dot{x}(t), \ddot{x}(t)) <= 0 \tag{2}$$

$$h_a(x(t), \dot{x}(t), \ddot{x}(t)) = 0\ . \tag{3}$$

if feasible, and $+\infty$ if the constraints cannot be satisfied.

In typical trajectory optimization, the optimization objective is a sum of cost terms along the whole trajectory. In our setting, we generalize this to a sum of cost terms for each ego-action edge in the tree, weighted by the probability of reaching this edge. The probability of reaching an edge is given by the decision model of the other agents.

C. Model of the other agents

Planning is performed by hypothesizing the decisions of the other agents and optimizing their eventual trajectories. Examples of models are:

- Adversarial: Other agents take decisions that maximize the trajectory costs of the ego-agent.

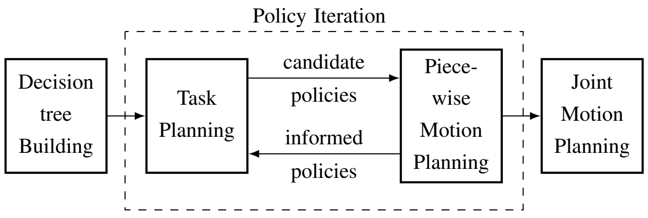

Fig. 3: TAMP algorithm

- Cooperative: Other agents minimize the trajectory costs of the ego-agent.
- Unpredictable: Other agents may take any action.

A policy π and a model of the other agents define the probability of reaching an edge a in the decision tree. We note this probability $p(a|\pi)$.

D. Optimal Policy and Trajectory Tree

We can now define the problem as finding a symbolic policy π and a trajectory tree ψ that minimize the discounted expected cost,

$$\Pi^* = \underset{(\pi,\psi)}{\operatorname{argmin}} \sum_{a \in \pi} p(a|\pi)\ \gamma^{k(a)}\ c(a, \psi(a))\ . \tag{4}$$

III. Solver

We propose a solver that works in three stages, schematized on Fig. 3. First, the decision tree is build. Second, we alternate Value Iteration and piece-wise trajectory optimization to compute the symbolic policy π^* jointly with a set of trajectory pieces. These pieces do not yet form a globally optimal trajectory tree, but inform the task planning about the cost associated with actions. In the third stage we fix π^* and optimize the full trajectory tree jointly.

A. Decision tree Building

The decision tree, is expanded from the start state using a breadth first strategy. We limit the tree size by expanding only to a certain maximal depth. In our current implementation, the problem is specified in a custom-made format. However, the environments and problems used so far can be described using a multi-agent extensions of the PPDL format. A recent extension with an open-source parser can be found here : `https://github.com/aig-upf/universal-pddl-parser-multiagent`.

B. Task Planning

We assume that at any point in time we have cost estimates $c(a)$ for each action a in the decision tree. These costs are initially all initialized with heuristic values. At each iteration, the optimal policy π^*, is computed from the tree using Value Iteration. The definition of the Bellman operator is specific for each agent-model and will be further described in future work. Then, more precise cost estimates of the actions of π^* are computed using piece-wise trajectory optimization, as described below.

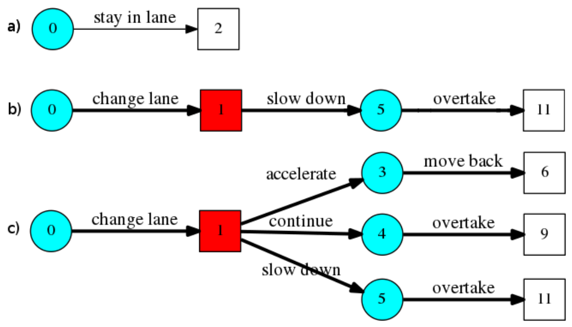

Fig. 4: Examples of policies. The policies a), b) and c) are obtained hypothesizing respectively an adversarial, cooperative and unpredictable other agent.

C. Piece-wise Trajectory Optimization

For each action of π^* we compute an estimate of trajectory cost $c(a)$ (if it was not already computed in previous iterations). For the sake of computational efficiency, we first perform a feasibility check by optimizing key-frames only (robot pose at each node). If feasible, we optimize the trajectory piece x, minimizing (1). The trajectory optimization methods are adopted from [1][2]. We save the resulting cost $c(a)$ and the trajectory piece x.

D. Joint Optimization of the Trajectory Tree

In the third stage of the solver, we fix the symbolic policy π^* found as described so far, and focus on the joint optimization of the trajectory tree ψ. So far we have only computed pieces x for each action edge. Concatenating these independently optimized pieces cannot capture long-term dependencies in the trajectories, e.g. when final actions influence earlier parts of the trajectory. The joint optimization of the trajectory tree leads to better and smoother motions. It is computationally costly, but performed only once for the best symbolic policy π^*.

We again solve the problem stage-wise. We first optimize all linear trajectories from the root node to each reached terminal nodes *independently*. Secondly, the trajectories are re-optimized with additional equality constraints enforcing that the common parts between trajectories are identical.

IV. Experimental Results

A. Overtaking behavior

We consider the overtaking problem introduced previously. Planning is performed using three different agents-models: cooperative, adversarial, and unpredictable, see Fig. 4. In the first two cases $a)$ and $b)$, the behavior of the other agent is purely deterministic leading to sequential policies. In $c)$, the policy is tree-like since the other agent may take any action. The ego car starts overtaking but moves back if the car in front accelerates, otherwise, the maneuver is pursued. For viewing the optimized trajectories, we refer the reader to the following video: `https://youtu.be/y-iLmFbwaFs`.

References

[1] M. Toussaint and M. Lopes, Multi-Bound Tree Search for Logic-Geometric Programming in Cooperative Manipulation Domains. Accepted at ICRA 2017.

[2] M. Toussaint, Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning. In Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI 2015), 2015.

[3] C. Phiquepal and M. Toussaint, Combined Task and Motion Planning under Partial Observability: An Optimization-Based Approach. RSS Workshop on Integrated Task and Motion Planning, 2017.