1: MAPF-R problem, indentification of aspects deserving to be visualized

----

Multi-Agent Pathfinding problem deals with finding paths for multiple agents
so they can avoid collisions with each other and reach their goal destination.

The majority of MAPF problems is solved on grids with discrete time, where the duration
of every action is one time step and each agent occupies exactly one single location in every time step. All agents are considered to be of the same shape and size and have the same constant speed. It uses space-time maps to describe agent's location at a certain moment of time - Cell(x, y, t), where (x,y) are space coordinates on the map and (t) is a timestep. Those discrete solutions simplify MAPF problem and are less applicable in real world within real time situations.

MAPF-R is Multi-Agent Pathfinding with continuous time, where agent motion is planned for a certain time interval. This time interval is a minium time considered to be safe for planned actions, which means there would be no collisions during this time interval. Agent's actions could be Move, which is moving to the next position, or Wait, which means that agent is keeping its position during the time interval. As an input we have workspace and agents parameters. Workspace is a 2D space, which can be represented as a graph G = (V, E), where vertices V are location, that agents can occupy, and edges E are line trajectories, which agents move along. Agents parameters are start location and goal destination. An output of algorithm is a MAPF-R problem solution, which is a joint plan for agents. Joint plan consists of individual plans for each agent, where a plan is sequence of actions for agent needed to be taken so it could reach its goal destination. The solution to MAPF-R problem should be cost-optimal, where cost of plan is a sum of durations of time-steps.

Although there are several algorithms for solving MAPF-R problem, algorithm, that is considered the most efficient, is CCBS, which stands for Continuous-time conflict based search.
[table with algorithm efficiency comparison]
First, CBS algorithm in discrete time should be de described. As an input we have workspace and agents parameters. Conflict-based search algorithm finds separate plans for each agent, then it determines conflicts between those plans and solves it by replannig whith specific constraints, that has been put on individual plans. Two types of conflicts are being considered in CBS: an edge conflict and a vertex conflict. A vertex conflict is a situation <i, j, v, t>, when agents i and j are planning to occupy vertex v at the same moment of time t. An edge conflict is a situation <i, j, e, t>, when agents i and j are planning to traverse edge e at the same time t from oposite directions. There is two types of constraints in conflict-based search algorithm, which are vertex constraint and edge constraint, Vertex constraint <i, v, t> is a configuration when agent i is not allowed to occupy vertex v at the time t. Similarly to vertex constraint, edge constraint <i, e, t> is a configuration when i is prohibited to traverse edge e at the moment of time t. CBS consists of high-level search and low-level search. Low-level search is looking for individual paths for each agent with given set of constraints. High-level search is a constraint-tree, which is a binary tree with a node representing constraints and joint plan. For each node, low-level search finds a plan for agents with imposed constraints. In case plan does not have neither vertex nor edge conflicts, the solution has been found. Otherwise, high-level search continues with the node which has the lowest cost.
[high-level search CBS]

CCBS, conflict-based search algorithm in continuous time, is different to CBS in the following way: agents can have any geometric shape, different speed and acceleration and agent's actions can have any duration, a conflicts in CCBS can occur between agents traversing different edges. Conflict in CCBS is a conflict between actions, for instance,

a conflict configurtion <a[i], t[i], a[j], t[j]> means, that if agent i executes action a[i] during the time period t[i] and if agent j executes action a[j] during the time period t[j], then collision will happen between agent i and j.
High-level search is similar to its discrete version. However, CCBS low-level search differ from its discrete version.
[Low-level search with SIPP]

After description of MAPF-R problem has been given, we will proceed with visualization process of MAPF-R problem. The main challenge is to determine an acceptable level of physical abstraction, i.e. determine what aspects of problem should be visualized and what can be ignored.

Workspace is presented as a graph on 2D space, where nodes interconnected with edges mean that agent is physically allowed to traverse between them. Other physical parameters of space, such as lightning, air temperature, floor level, height of ceiling, can be ignored in visualization.

It was stated, that MAPF-R takes into account agents physical shapes and parameters, subsequently physical restrictions, which are implied, should be visualized properly. It will allow user to detect danger areas, where the risk of collision between agents is higher than average. Since agent's constraints are presented as a set of time intervals, agent have to move in real time according to its speed and acceleration. Visualization of real-time movement will allow user to indentify if agent could speed up or slow down in order to be more effective.