MULTI-AGENT SYSTEMS - LECTURE 3

OHAD LUTZKY

1. Multi-Entity Models

- Joint accomplishment of goals
- Multiple entity model

Given a multi-entity model, and a goal for each player, we'll want to find a joint plan to accomplish all goals. We'll have several kinds of questions:

- Can the length of the plan be bounded?
- In more specific models, is the computational problem solvable with efficient time/space usage?

The existence of a finite plan to achieve all goals (cooperatively) is provable. That is, we can prove that given a set of plans to achieve all goals exists, it is finite.

Proof sketch: First, we must notice that since the agents' plans are common knowledge, any learnt fact can be translated into knowledge about the initial configuration[1]. Without loss of generality, we may observe an agent's plan as a decision tree, where a fork is only performed when new knowledge is attained. If there is a situation for which along all trees (plans) there is no fork in any branch, along more than $|\mathcal{C}|$ [2] steps, then a shorter plan can be found. Therefore, because $|\mathcal{C}|$ and $|C_0|$ are finite, we have QED.

A rich model which is contained in the said model is the full-information model. In this case, simply $C_0 = c_0$. This is also called C-CGA (Complete information - Cooperative Goal Achievement). In this case, it can be shown that the problem of deciding whether there is a plan to achieve all goals can be computed in polynomial space. That is, we'll want to say whether such a joint plan exists.

Proof sketch: In such cases, we'll give a non-deterministic algorithm, that is - one which can make guesses, which can be stored in a polynomial space, so that under these guesses, we won't use more than polynomial space. In the initial configuration, we will guess what the joint action of the agents will be, and calculate what the next configuration will be. Now we'll guess what the next joint action would be, and so on. Notice that $|\mathcal{C}|$ guesses will suffice in order to achieve the goals (if such a way exists). Such an algorithm is in NPSPACE. We'll use a theorem which says that PSPACE = NPSPACE.

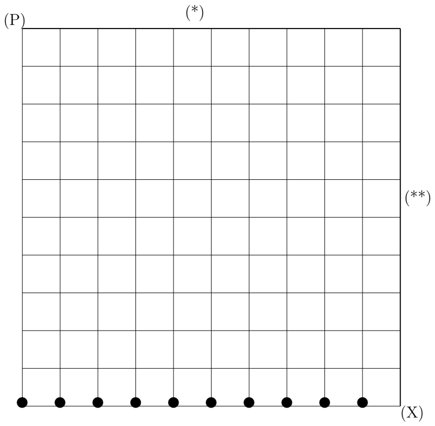
Another interesting case if F-CGA, which speaks of a case in which at most t participants can "fail". Meaning, in a certain situation (which is not known in advanced), a participant can perform a certain operation ("fail"), which is denoted by him being in a "failed" state, in which his action is Λ.

Assume that the agents can coordinate their actions and see the state. This creates a problem for us in the form of finding a strategy to "defeat" the failures. This is problematic, because the failure-option tree is exponential in size. However, instead of creating the entire tree, we can give an algorithm in NPSPACE, so that we go over all of the options, putting "no failure" as the last option.

---

[1] This is because the initial configuration is the only remaining unknown variable.

[2] $|\mathcal{C}|$ is the number of configurations in the basic model, and does not present knowledge states



2. Social Systems

What happens when the agents' objectives aren't known in advance, or when there isn't a central planner? We call these systems *Social Systems.*

So far we've shown:

1. Canonical display of knowledge by multi-entity models
2. In such systems, looking from the perspective of a central planner, it is enough to consider *finite plans*
3. Typical problems, within the context of finding solutions in trees, can be solved in PSPACE.

Essentially, we've looked at two opposing edges of the same problem - centralized solutions, and completely distributed ones. There's actually an infinite spectrum between those two, which is called *Social Systems.*

We will begin with an example. We have an $n \times n$ grid, and m robots ($m \leq n-1$, for simplicity). Motion on this grid is discrete. Furthermore, we make the assumptions of *no perception* and *no communication.* Our goals:

- Avoid collision (*safety*)
- Enable goal achievement (*liveness*)

The completely distributed and completely centralized models have an extremely hard time with problems like these.

A social law would be a motion law - a set of limitations on possible motion. Given the social law, any behaviour the agents exhibit which obeys the social law will lead to *safety.* If while behaving socially an agent has reached his goal, thus he has a social plan (which obeys the laws) which ensures achievement of his goal, *regardless* of what the other agents do, so long as they obey the social law.

One good example of a social law in this case would be a perpetual hamilton cycle - this will take all agents through all vertices in the grid, and preserve safety. However, this isn't a particularily interesting one, because it's extremely limiting - no freedom is given to the agents. Also, it's inefficient, because achieveing any goal is $O(n^2)$.

A more successful social law for this situation is this: Assume, for simplicity's sake, that all robots start at the lowest row, flush to the left. Mark the topmost row (*) and the rightmost row (**), the top-left corner (P), and let (X) be the bottom-right corner (see Figure 2). The agents will move to (X) and then choose (*freely*, but they will probably want to pass through their goal) some path to P, without moving on (*) or (**), where they must select one of the shortest possible paths. Returning to X is through (*) and (**). The goal is to achieve any selected location (except for the bottom row).

It's easy to see that, if there are no collisions, then one can get to any goal point in $O(n)$. To see why this law ensures *safety*, one must see that the distance from (P) stays constant.

2.1. **The Golden Mean Problem.** Assume we have a pair of agents, playing a game, with a set of strategies $S = s_1, \ldots, s_n$, and a set of goals $G = g_1, \ldots, g_n$.

$$\forall i \forall g \in G \exists s \in S u_i^g(s, \sigma_{-i}) \geq \varepsilon, \forall \sigma_{-i} \in \bar{S}^{n-1}$$

The case $n = 2$ demonstrates wel. Given a goal g_1 for agent one, then there are many ways to attain it, but that demands limiting agent 2. Symmetrically, limiting agent 2 will demand limiting agent 1, and suddenly agent 1 can no longer attain a different goal. Essentially, this means that one must seek a fixed point where the constraints aren't too limiting for oneself.

This problem is tightly linked with the SAT problem (satisfiability).