Multi Agent Resource Allocation: a Comparison of Five Negotiation Protocols

Daniela Briola
Viviana Mascardi

WOA 2011 @ UNICAL
The resource allocation problem we are considering (1)

- From Ansaldo STS case study
- Resources are the “platforms (nodes) inside a station”:
  - indivisible
  - non-sharable
- Resources may become unavailable
  - they break up and become available again (when fixed)
- Organized in a directed graph of dependencies (because connected by railway tracks), represented as arcs $R_1 \rightarrow R$, $R_2 \rightarrow R$, ..., $R_n \rightarrow R$ in the graph
- To use $R$, an entity must use one resource among ($R_1$, $R_2$... $R_n$)
The model of the problem

- A directed and non-planar graph that entities must traverse from one start point to one end point
- “Start points” and “End points” resources (A and H)
- A set of couples of conflicting arcs in the graph of dependencies (C->D, F->G)
  - The usage of C->D during TS makes C-> and F->G occupied for TS
- Continuous and linear time model
The resource allocation problem we are considering (2)

- A set of entities (trains) with
  - different priorities
  - needing to use some of the available resources for a predefined time in a predefined time interval
  - a static allocation plan that selects resources in predefined time slots
- They enter and exit the system in any moment
- They can change the chosen resources inside the graph, but never the first and the last resource of the plan
- They can change the resources time usage (duration of usage and interval of usage)
The FYPA (Find Your Path, Agent!) Negotiation protocol

- It simulates the real world
  - Trains may use nodes for longer than planned
  - Nodes can break up
- The protocol looks for a dynamic re-allocation of the resources to the entities such that the re-allocation
  - is free of conflicts
  - is completed within a pre-defined amount of time
- Minimizes as much as possible (sub-optimal solution):
  - the changes between the new plan and the static allocation
  - the delay of the entities in reaching the end point
  - the number of entities and resources involved in the re-allocation process
The FYPA protocol

- Two agent’s roles: Train Agents and Node Agents
- Every conflict is solved with an implicit negotiation between trains
- To use a resource, every train (T1) must reserve it
  - If the resource is free, no problems arise
  - If the resource is already reserved (by T2), the train acts considering its priority
- Every train uses priority to decide how to act:
  - If T1 priority is lower than T2 priority, than wait or look for another path
  - If T1 priority is higher than T2 priority, “steal the reservation”
    - The problem is than managed by T2
The other considered negotiation protocols

- We present four protocols similar to FYPA that we found in literature
- MPCA (Multi-Party Collision Avoidance) for airplane collision avoidance (D. Šišlák, J. Samek, and M. Pěchouček)
- APR (Airplane rerouting) for airplane collision avoidance (A. Agogino and K. Tumer)
- Waypoints for autonomous robots collision avoidance (O. Purwin, R. D’Andrea, and J.-W. Lee)
- SPAM (Scalable Protocol for Anytime Multi-level) for target tracking with sensors (R. Mailler, V. Lesser, and B. Horling)
MPCA
(Multi-Party Collision Avoidance)
Why considering MPCA?

- Area: airspace management (avoiding collisions among aerial vehicles)
- A group of autonomous airplanes with a mission need to coordinate themselves to avoid collisions
- A mission is made of several points that must be reached in a specified time interval
- Every plane must maintain a minimum distance from all the others, and there are no-flight zones in the airspace
- When two, or more, airplanes have a part of the plan in common, they need to change it to avoid crashing
  - This domain is very similar to the FYPA one
MPCA protocol

- Every agent can only interact with the ones within a range $R$ defined at the start of the simulation: these are the other planes that it can “see” (on a virtual radar) from its position in the space.
- Every agent sends to the others it sees an update on its future mission steps so they can check if there are conflicts.
- The authors propose two solutions to the above problem:
  - a local one
  - a global one
The local solution

- “Iterative peer-to-peer collision avoidance (IPPCA)”
- The two agents involved in a conflict propose a list of possible changes to their path and look if they can adopt one of these change.
- If they are not able to find a solution, they generate more maneuvers (accepting higher values for the parameters) and add them to the list: then they try again to solve the conflict.
- They repeat this procedure until they find a solution.
- In this way the list can be filled with tens or hundreds maneuvers and the solution is always found.
- When the agents have solved their conflict, they will check if other collisions exist and will start the protocol again to solve the first one that arises.
- If an agent is involved in more than one collision, it will solve the one that is expected to occur first in time.
The global solution

- This is the MPCA algorithm
- The agents that have a colliding path create a group: they try to change their paths and add to the group those agents that could be interested by these path changes.
- The group is enlarged to involve all the agents that are near to the ones that have a conflict.
- Then, the possible alternative plans are analyzed by the group till a solution is found, and all the agents will modify their path as decided.
- The group searches the states space of possible plan changes using an A* algorithm.
- To simplify the communication and synchronization issues a coordinator agent for each group is created: it collects all the information it needs to solve the problem and than it finds a solution.
- If an agent is involved in more multi-party groups, it will only join the one with the earliest expected collision in the time line.
Why not using MPCA in our case study?

- The MPCA algorithm in particular is very similar to the FYPA one because it is based on the same idea of “moving others to get space for you”.

- Some aspects of the MPCA domain are quite different from ours because airplanes have much more flexibility in their movements than trains.
  - they must avoid certain zones and avoid other agents, but must neither follow rigid and limited paths, nor have to reach a fixed point but and area.
  - they move in a 3D space.
AirPlane Rerouting (APR)
Why considering APR?

- Area: airspace management (avoiding collisions among aerial vehicles)
- US airspace: the space is divided into regional centers and again into sectors
- The algorithm uses a global evaluation function that considers the congestion in a particular set of sectors and the global air traffic delay.
- Using this common function agents independently take decisions about how changing their plans.
- In this system agents are ground location throughout the airspace and are called “fixes”.
- Each agent is responsible for the aircrafts going through its fix.
- Every airplane has a “flight plan” consisting of a sequence of fixes.
- This domain representation and the agents organization is quite similar to the FYPA one
Agents can change the plan of the interested airplanes in three ways:

- Miles in trail (MIT): agents control the distance that the airplane must keep from each other while approaching a fix. If the MIT values is high, fewer planes will be able to cross this area because they need to slow down their velocity to maintain the distance.

- Ground delays: an agent can control how long aircrafts that will eventually go through a fix should wait on the ground, that is, the airplanes will arrive later at the fix.

- Rerouting: an agent can divert the foreseen planes of its fix making them choosing another path.

A sets of agents that can influence themselves rerouting airplanes in their fixes is identified: each agent lists the possible solutions to the congestion problem and then chooses the best solution using different learning algorithms.
Why not using APR in our case study?

- This algorithm is interesting but seems to work only if the groups are limited to few agents, because it is based on the list of possible choices, that becomes too long if there are large sets of agents, or too many possibilities of plan changes.
- The agents seem not to negotiate, but only to work independently using the same evaluation strategy and only one possible change to the airplane route.
- All these constraints seem prevent the system from being applicable to very complex scenarios or short term simulations (as our scenario).
WayPoints
Why considering WAYPOINTS?

- Area: autonomous robots able to independently move and need to avoid collisions while trying to reach a desired destination.

- For the algorithm the following assumptions are made:
  - all computation/control is done on board;
  - the total number of agents is known;
  - motion primitives are available to move the agents in a deterministic fashion;
  - point-to-point communication between agents is supported;
  - agents can localize themselves, but not others.

- The algorithm is executed considering discrete time (agents work with “frames”, that is, they assume a discrete divisions of time).

- The domain is different in some aspects from the FYPA one, but could be mapped in it. The protocol shows some interesting similarities with FYPA.
WAYPOINTS protocol

- Every agent is trying to reach its task location
- Position and velocity of the agent are expressed in a global Cartesian coordinate system
- Every agent reserves an exclusive area (called A area) for itself and always remains inside that area, and no two reserved areas are allowed to intersect at any time.
- They do not choose the reserved area (A) directly: agents are indicating their intentions by requesting an area (called B area) first, and exchanging it with the others.
- Agents can change B arbitrarily. However, significant changes to B can cause the negotiation cycle to start over.
- The requested area B has to contain the reserved area A at all times
  - an agent cannot move to a location that is not inside the requested area
Two intersecting B areas indicate a possible conflict.

The agents will negotiate to find out which one gets priority and how the A areas are being selected.

The base for this negotiation is a scalar cost function:
- the agents with a conflict exchange their respective costs and choose who has the higher priority using this, choosing also who will change its A area.
- The agent with the lower priority will reduce it’s A area (stopping and waiting for the other agent to pass over).

Instead of stopping, an agent can decide to change its path to avoid an obstacle (or simply a point of its paths that intersects with many other agents) and to move around that.

It will choose a “way-point”, that is a new intermediate destination, and will try to reach it using the same algorithm shown above.

Then it will start again to reach the initial destination.
Why not using WayPoints in our case study?

- This algorithm is similar to FYPA in the idea of how the agents collaborate to solve the conflicts and how they can change their strategy: in both protocols agents can stop and wait for the other to move, or can change their path.
- The difference is that in our domain the paths are limited to a predefined set and are divided into fixed parts.
- In FYPA, every subpart is managed by a Resource agent, whereas in that article agents are able to move without limitations and without intermediate agents.
- The agents operate in a wireless environment, so the number of exchanged messages and the real distance of the agents can make the difference on the behavior of the entities: in this case some limitations and heuristics must be adopted.
  - in FYPA the only limitation is due to the computational time of managing all the messages.
SPAM
(Scalable Protocol for Anytime Multi-level)
Why considering SPAM?

- **Area:** sensors platforms and constraint satisfaction problem
- **Agents** are interested in using a set of limited resources in different moments for different periods: the resources are three sensors platforms, and agents need at least three sensors to track an entity moving in the environment.
- More sensors give a more accurate target’s location.
- Each platform is managed by a track manager, which is also in charge of localizing and following a target, deciding which sensors (of which platform) it needs to do this and when
- This agent organization is similar to the FYPA one
SPAM protocol

- Every agent has an utility function $U$ that uses to evaluate the proposed solution (the set of sensors and the period of usage).
- SPAM protocol works in two main phases
- Stage 1 tries to find a solution within the context of the information that the protocol has when it starts up.
- The protocol attempts to maximize the social utility.
- Each of the agents tries to maximize its local utility without causing new constraint violations. If this can be done, then no further negotiation is necessary, and the protocol terminates at the end of stage 1.
- The second function of stage 1 is to ensure that some utility is obtained while waiting for stage 2 to complete. A temporary solution is applied while the mediator tries to get a better solution.
If stage 1 was activated because of a newly discovered conflict, and a conflict-free solution cannot be found, the manager just enters stage 2.

Stage 2 attempts to solve all local conflicts that a track manager has by elevating the negotiation to the track managers that are in direct conflict over the desired resources.

The originating track manager takes the role of the negotiation mediator and starts collecting all the information it needs to generate alternative solutions.

These solutions are generated without a global vision, so they are conflict free only from the point of view of the mediator.

If the chosen solution will cause other conflicts then other agents will try to solve them, even starting again all the algorithm.

The mediator collects information from all the other involved agents, calculates a list of possible solutions and sends them to the agents. Then it receives back these lists reordered by every agents.

The mediator chooses a solution, possibly a good one for all the agents involved, and sends it to the agents that must apply it.
Why not using SPAM in our case study?

- Comparing this protocol to FYPA, the main difference that emerges is the possibility for a track manager agent to “loose a track”, namely give up tracking an object, if it becomes too hard to do that.
  - In SPAM this event is allowed, even if it is the last option, whereas in our protocol it is not possible for a train to give up obtaining resources!

- Furthermore, the representation of the domain as a constraint problem should be almost difficult for the FYPA domain.

- Finally we preferred a real distributed negotiation, while the solution proposed here is partially centralized. In a way similar to the Contract Net protocol, agents do not negotiate, they choose one of them to solve the constraint problem and then apply the solution it proposes.
Comparison
## Accepted agent definition

<table>
<thead>
<tr>
<th></th>
<th>Autonomous</th>
<th>Situated</th>
<th>Reactive</th>
<th>Pro-active</th>
<th>Social</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPCA</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>APR</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>LIMITED</td>
<td>LIMITED</td>
</tr>
<tr>
<td>WP</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>LIMITED</td>
</tr>
<tr>
<td>SPAM</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>LIMITED</td>
</tr>
<tr>
<td>FYPA</td>
<td>LIMITED</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
## Purpose and Approach of the MAS

<table>
<thead>
<tr>
<th></th>
<th>Simulation</th>
<th>Real control</th>
<th>Approach</th>
<th>Industrial</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPCA</strong></td>
<td>YES</td>
<td>YES</td>
<td>A* state search</td>
<td>BAE Systems and others</td>
</tr>
<tr>
<td><strong>APR</strong></td>
<td>YES</td>
<td>LIMITED</td>
<td>Learning algorithm</td>
<td>Foreseen for the Federal Flight Administration</td>
</tr>
<tr>
<td><strong>WP</strong></td>
<td>NO</td>
<td>YES</td>
<td>Scalar cost Function</td>
<td>NO</td>
</tr>
<tr>
<td><strong>SPAM</strong></td>
<td>YES</td>
<td>NO</td>
<td>Distributed Constraints Satisfaction Problem</td>
<td>NO</td>
</tr>
<tr>
<td><strong>FYPA</strong></td>
<td>YES</td>
<td>LIMITED</td>
<td>Distributed Resource Allocation</td>
<td>Ansaldo STS</td>
</tr>
<tr>
<td>Implementation features</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td><strong>Platform</strong></td>
<td><strong>Var. Strat.</strong></td>
<td><strong>Code available</strong></td>
<td><strong>User</strong></td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>MPCA</td>
<td>JAVA</td>
<td>AGENTFLY/AGLOBE</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>APR</td>
<td>UNK</td>
<td>FACET</td>
<td>NO</td>
<td>UNK</td>
</tr>
<tr>
<td>WP</td>
<td>C++</td>
<td>RoboCup</td>
<td>YES</td>
<td>FREE</td>
</tr>
<tr>
<td>SPAM</td>
<td>JAVA</td>
<td>FARM</td>
<td>YES</td>
<td>UNK</td>
</tr>
<tr>
<td>FYPA</td>
<td>JAVA</td>
<td>JADE</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>
Thank you!!

Any questions?