

G54DIA: Designing Intelligent Agents

Lecture 12: Multi-Agent Systems II

Brian Logan

School of Computer Science

bsl@cs.nott.ac.uk

Outline of this lecture

- teamwork
- joint intentions theory
- teamwork models
- example: STEAM

Multi-agent systems

- a *multi-agent system* is a system in which several agents share a common task environment and *cooperate* at least part of the time
- the *agents* can have any of the architectures we have seen so far, e.g., reactive or deliberative or hybrid
- all the agents may have the same architecture or they may have different architectures
- the *environment* may not appear the same to the agents if they are different, e.g., if they have different sensors and actions

Co-operation in multi-agent systems

- agents are *self-interested* and do not share a common goal
 - e.g., they are designed to represent the interests of different individuals or organisations
 - agents co-operate because it helps them achieve their own goals
- agents implicitly or explicitly share a *common goal*
 - benevolently work to achieve the overall objectives of the system, even when these conflict with the agent's own goals
 - e.g., when the agents are 'owned' by the same organisation or individual

Shared goals

- we will focus on the special case in which all the agents in the MAS cooperate to achieve one or more system or *organisational goals*
- the agents co-operate to perform some task that a single agent can't do on its own
 - because a single agent doesn't have all the capabilities or knowledge required to perform the task
 - because a single agent would be too slow
- note that there may still be elements of competition, e.g., if the agents compete for the organisation's resources
- mechanisms are still required to ensure that resources and tasks are allocated appropriately

Task sharing

- *task sharing* is the problem of determining how tasks are allocated to individual agents in a multi-agent system
- for homogeneous (e.g., totipotent) agents this is straightforward—only concern is load balancing
- if the agents are heterogeneous (have differing capabilities) and/or are autonomous (can refuse tasks), then task sharing involves reaching agreements between agents

Contract net protocol

- *contract net protocol* is a way of achieving efficient co-operation through task sharing in networks of (possibly heterogeneous, autonomous) agents
 - *task announcement*: an agent which generates (or receives) a task broadcasts a description of the task to some or all of the agents
 - *bid response*: agents respond to the task announcement with a bid
 - *task allocation*: the agent which announced the task allocates it to one or more of the bidding agents
 - *expediting*: the agent to which the task was allocated carries it out

Agent teams

- for many complex, dynamic multi-agent domains, e.g.,
 - virtual training
 - entertainment
 - information integration
 - robotics etc.
- more complex models of *teamwork* are required

Teamwork

- teamwork involves co-operation between agents to achieve a common goal
- team members have differing, incomplete, inconsistent views of their environment
- team members may fail to fulfil their responsibilities or discover unexpected opportunities
- team members should respond appropriately to failures and opportunities

Examples of teamwork failures

... from a military simulation which included *pilot agents* for a company of synthetic attack helicopters:

- on abnormal termination of an engagement, the company commander returned to base alone, abandoning members of its own company at the battle position
- on reaching the holding area the company waited while a single scout went to scout the battle position; the scout crashed into a hillside and the rest of the company waited indefinitely for the scout's report

More examples of teamwork failures

- only a scout made it to the holding area (all the other helicopters crashed or were shot down), but the scout scouted the battle position anyway, and waited indefinitely for its non-existent company to move forward
- when all the members of the company ran out of ammunition, the company failed to infer that their mission could not continue

Avoiding teamwork failures

- one way to overcome these problems is to keep adding domain-specific coordination plans
 - however without a framework which allows prediction of teamwork failures, coordination plans have to be added on a case by case basis after the failure has been encountered in an actual run
 - for complex domains, a large number of coordination plans are required
 - difficult to reuse the resulting plans in other domains
- solution: explicit model of teamwork based on *joint intentions theory*

Joint intentions

- a team of agents co-operating to achieve a goal must have:
 - a *joint commitment* to the (team) goal; and
 - *individual commitments* to the specific tasks they have been assigned
- joint commitment is *distributed* between the team members – requires rules or protocols for:
 - how an agent should behave towards its fellow team members while the joint commitment is in force
 - the conditions under which a joint commitment can be abandoned

Joint intentions theory (Cohen & Levesque)

- a team T *jointly intends* a team action if team members are jointly committed to completing that team action, while mutually believing that they are doing so
- a *joint commitment* is defined as *joint persistent goal*, $JPG(T, p, q)$, where p denotes the completion of a team action, and q is an (ir)relevance condition, which allows the team to drop the JPG if they mutually believe that q is false
- to enter into a joint commitment, all team members must establish appropriate *mutual beliefs* and *commitments* via an explicit exchange of *request* and *confirm* (or *refuse*) speech acts

Joint persistent goals

- a joint persistent goal $JPG(T, p, q)$ holds iff:
 - all team members mutually believe that p is currently false
 - all team members have p as their mutual goal, i.e., they mutually know that they want p to be eventually true
 - all team members mutually believe that until p is known to be achieved, unachievable or irrelevant, they mutually believe that they each hold p as a *weak achievement goal* $WAG(m, p, T, q)$

Weak achievement goals

- $WAG(m, p, T, q)$ where m is a team member in T , implies that one of the following holds:
 - m believes p is currently false and wants it to be eventually true, i.e., p is a normal achievement goal; or
 - having privately discovered p to be achieved, unachievable, or irrelevant (because q is false), m has committed to having this private belief become T 's mutual belief
 - this typically involves m communicating with its team mates about the status of p

Mutual belief

- mutual belief means that “all agents believe that p , and all agents believe that all agents believe that p ”
- it ensures that team members are updated about the status of team activities
- the commitment to attain *mutual belief* in the termination of p is a key aspect of *JPG*
- if a team member, m , privately believes that p has terminated (for whatever reason), *JPG*(T , p , q) is dissolved, but m is left with a commitment to have its private belief become T 's mutual belief

Teamwork models

- each agent has a *general model* of teamwork
 - agents *autonomously* reason about their coordination and communication responsibilities using the model
 - allows each individual agent to anticipate and avoid (or recover from) teamwork failures
- teamwork models should be reusable across domains

Example: STEAM

- STEAM (Shell for TEAMwork) is a teamwork model based on joint intentions theory developed for SOAR agents (Tambe 1997)
- teamwork is achieved by agents building up a partial hierarchy of joint intentions
- agents monitor the team's and individual member's performance to detect and recover from failures
- to reduce the communication overhead of teamwork, a decision theoretic model is used to determine when to communicate information to other members of the team

STEAM teams

- teams can be flat or hierarchically organised
- each team defines a set of *roles* which can be assigned to individuals or sub-teams
- roles may be either persistent or assigned on a short term basis, and be pre-assigned or dynamically reassigned during plan execution
- team activities are represented by *team operators* (reactive plans)
- each operator specifies a particular task and how these can be broken down into subtasks
- team operators evaluated relative to *team state memory* which stores the (agent's view of) the mutual beliefs of the team

STEAM teams

- team organisation is separate from task decomposition specified by team operators
- mapping from team organisation to tasks is via roles
- roles constrain which sub-tasks of the current team operator an agent can adopt
- an agent can only perform those tasks which are consistent with its current role within the team

Implementation

- three categories of domain independent rules:
- coherence preserving rules which ensure that agents communicate to establish mutual belief of relevant conditions (e.g., a plan becoming unachievable)
- monitor and repair rules which specify how team members can be replaced if they fail to achieve their tasks
- communication rules which evaluate the utility of communication (to avoid redundant communication)

Applications of STEAM

- STEAM has been applied to a (small) number of teamwork problems involving teams of up to 16 agents, e.g.:
- military training simulations:
 - pilot agents for a company of 8 synthetic attack helicopters
 - pilot agents for 4-12 transport helicopters protected by 2-4 escort helicopters
- RoboCup: coordinating play in the CMUnited RoboCup team

Architectural implications of teamwork

- teamwork models require modifications to the underlying agent architecture
- typically requires explicit representations of mutual beliefs, team plans, and team goals
- for STEAM this involved:
 - changes to the SOAR architecture
 - about 300 (mostly) domain-independent SOAR rules which implement the teamwork model
- however for typical SOAR agents, the overhead of the teamwork model is fairly small

The rest of the module

- lecture tomorrow (6th of March) on the second coursework
- group tutorial next week (10th of March, 9-10)
- individual tutorials: 10th of March, 11-12 and during all remaining lecture slots apart from 17th of March, 9-10
- a lecture with feedback on coursework 1 and SET/SEM (17th of March, 9-10)
- **cw2 submission** Monday 30 March 23:55