CHAPTER 9: WORKING TOGETHER

An Introduction to Multiagent Systems

http://www.csc.liv.ac.uk/~mjw/pubs/imas/

1 Working Together

- Why and how to agents work together?
- Important to make a distinction between:
  - *benevolent agents* and
  - *self-interested agents*.

1.1 Benevolent Agents

- If we “own” the whole system, we can design agents to help each other whenever asked.
- In this case, we can assume agents are *benevolent*: our best interest is their best interest.
- Problem-solving in benevolent systems is *cooperative distributed problem solving* (CDPS).
- *Benevolence simplifies the system design task enormously!*

1.2 Self-Interested Agents

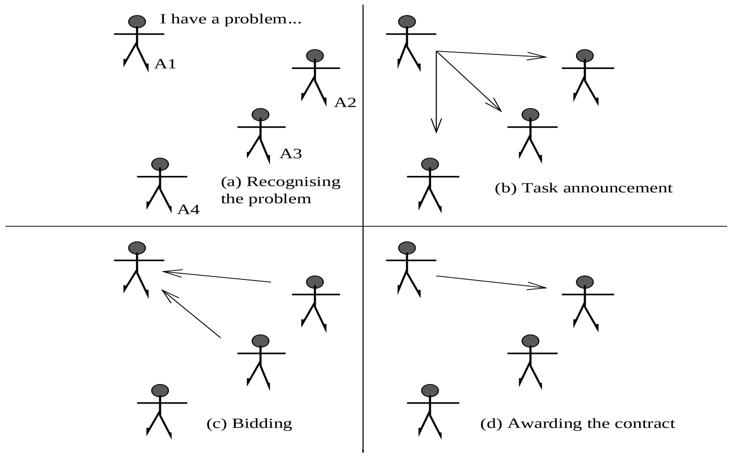
- If agents represent individuals or organisations, (the more general case), then we cannot make the benevolence assumption:
- Agents will be assumed to act to further there own interests, possibly at expense of others.
- Potential for *conflict*.
- May complicate the design task enormously.

2 Task Sharing and Result Sharing

- Two main modes of cooperative problem solving:
  - *task sharing*:
    components of a task are distributed to component agents;
  - *result sharing*:
    information (partial results etc) is distributed.

3 The Contract Net

- Well known task-sharing protocol for *task allocation* is *contract net*:
  1. Recognition;
  2. Announcement;
  3. Bidding;
  4. Awarding;
  5. Expediting.



3.1 Recognition

- In this stage, an agent recognises it has a problem it wants help with.

  Agent has a goal, and either. . .

  – realises it cannot achieve the goal in isolation — does not have capability;

  – realises it would prefer not to achieve the goal in isolation (typically because of solution quality, deadline, etc)

3.2 Announcement

- In this stage, the agent with the task sends out an *announcement* of the task which includes a *specification* of the task to be achieved.
- Specification must encode:
  - description of task itself (maybe executable);
  - any constraints (e.g., deadlines, quality constraints).
  - meta-task information (e.g., "bids must be submitted by...")
- The announcement is then *broadcast*.

3.3 Bidding

- Agents that receive the announcement decide for themselves whether they wish to *bid* for the task.
- Factors:
  - agent must decide whether it is capable of expediting task;
  - agent must determine quality constraints & price information (if relevant).
- If they do choose to bid, then they submit a *tender*.

3.4 Awarding & Expediting

- Agent that sent task announcement must choose between bids & decide who to "award the contract" to.
- The result of this process is communicated to agents that submitted a bid.
- The successful *contractor* then expedites the task.
- May involve generating further manager-contractor relationships: *sub-contracting*.

3.5 Issues for Implementing Contract Net

- How to. . .
  - . . . specify *tasks*?
  - . . . specify *quality of service*?
  - . . . select between competing offers?
  - . . . differentiate between offers based on multiple criteria?

4 Result Sharing in Blackboard Systems

- The first scheme for cooperative problem solving: the *blackboard system*.
- Results shared via shared data structure (BB).
- Multiple agents (KSs/KAs) can read and write to BB.
- Agents write partial solutions to BB.
- BB may be structured into hierarchy.
- Mutual exclusion over BB required ⇒ bottleneck.
- Not concurrent activity.
- Compare: LINDA tuple spaces, JAVASPACES.

5 Result Sharing in Subscribe/Notify Pattern

- Common design pattern in OO systems: *subscribe/notify*.
- An object *subscribes* to another object, saying “tell me when event e happens”.
- When event e happens, original object is notified.
- Information pro-actively *shared* between objects.
- Objects required to know about the *interests* of other objects ⇒ inform objects when relevant information arises.