LECTURE 4

SUMMARY

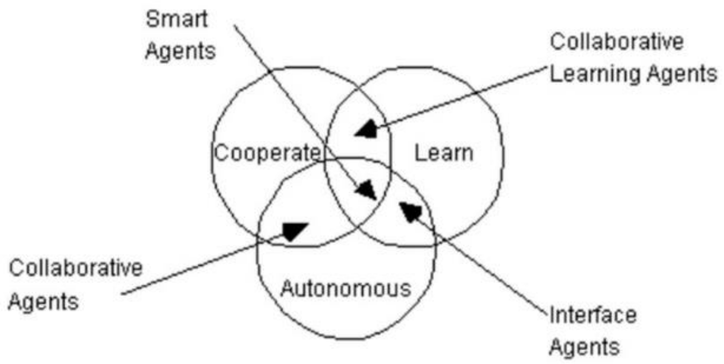
1. Multiagent systems

- a **multiagent system** (MAS) is one that consists of a number of agents, which interact with one another
- a MAS distributes computational resources and capabilities across a network of interconnected agents
- a MAS is **descentralized**
- in the most general case, agents will be acting on behalf of users with different goals and motivation. To successfully interact, they will require the ability to
  - **cooperate**
  - **coordinate**
  - **negotiate**

  with each other, much as people do.
- can support distributed collaborative problem solving by agent collections that dynamically organize themselves.
- support a modular, extensible approach to design of complex information systems.

- important issues in MAS
  - inter-agent **communication**
    - of knowledge, intentions, beliefs
  - inter-agent **collaboration**
    - through negotiation among self-interested rational agents
  - **coordination** and **control**

- **collaborative agents**
  - emphasize **autonomy** and **cooperation**
  - in order to perform tasks of their owners
  - they may **learn**
  - are capable of acting **rationally** and **autonomously**



- inter-agent **cooperation** can be achieved through:
    - grouping
    - communication
    - specialization
    - sharing tasks and resources
    - coordination of actions
    - conflict resolution by arbitration and negotiation
- **MAS organizations**
    - MAS organizations are defined by an assembly of classes of agents characterized by their assigned roles and set of abstract relationships among the roles.
        - *acquaintance*
        - *subordination*
        - *conflict*.
    - possible **approaches** for MAS organizations
        - **horizontal modular**
            - different functional components are separated from one another
            - a technique adopted for SE
            - horizontal design
            - architectures:
                - Prodigy, 1991
                    - planning, learning
                - ICARUS
                    - 1998, cognitive architecture
                    - evolution of original architecture
        - **hierarchical**
            - vertical
        - **societies governed by conventions**
            - negotiations protocols
        - **ant colonies**
            - simple rules of interaction
            - MAS, RL
        - **immune systems**
        - **evolution**
        - **…**
    - Organizational paradigms
    - **MOISE** organizational model for MAS
        - http://moise.sourceforge.net/
        - MOISE Framework

        - Organization Oriented Programming of MAS

The two key problems in a MAS

- **agent design** (the micro perspective)
    - *how do we build agents that are capable of independent autonomous action in order to successfully carry out the tasks that we delegate to them*?
- **society design** (the macro perspective)
    - *how do we build agents that are capable of interacting (cooperating, coordinating, negotiating) autonomous action in order to successfully carry out the tasks that we delegate to them*?

Some views of the MAS field

- **agents** as a paradigm for **software engineering**
- **agents** as tools for understanding human (natural) **societies**
    - MAS provide tools for simulating societies, which my help shed some light on various kind of social processes.
    - E.g.
        - simulation of biological and natural systems
        - simulation of natural disasters (meteorology)
        - ecological models
        - socio-ecological systems
        - ecosystems management

Advantages of a MAS

1. It is descentralized and does not suffer from the "*single point of failure problem*" associated with centralized systems.
2. Models problems in terms of autonomous interacting component agents, which is proving to be a more natural way of representing various tasks, such as:
    - task allocation, team planning, user preferences, open environments, a.s.o
3. Provides solutions in situations where data sources and expertise is spatially and/or temporally distributed.
    - **classical view**
        - data are distributed
    - **another view**
        - data are not spatially distributed
        - multiple agents are used for increasing performance
4. Increases system performance
    - computational efficiency
    - robustness
    - maintainability
    - flexibility

Applications of MAS research

- agents are indicated for domains where ***flexible autonomous*** action is required
    - agents are often implemented as
        - *processes* (UNIX, C, Python)
        - *threads* (Java)

- MAS are indicated for domains where
  - control, data, expertise are distributed
    - e.g. product design, distributed health-care
  - centralized control is impractical or impossible
  - processing nodes have competing/conflicting viewpoints or objectives
- MAS applications cover a variety of domains, including:
  - **robotics** (e.g. multirobot patrolling, multirobot exploration, etc)
    - **multirobot systems (MRP)**
  - **aircraft maintenance**
  - **wireless collaboration and communication**
  - **military logistics planning, defense systems**
  - **supply chain management**
  - **auctioning, trading, BPM, e-business**
  - **geographic information systems (GIS)**
  - **networking and mobile technology**
  - **….**

Example: agents in e-banking

- **agent** – **personal financial assistant**
- agent roles
- electronic payment services

2. Distributed problem solving and planning ([1], Chapter 3)

Distributed problem solving (DPS)

- DPS is a subfield of **Distributed Artificial Intelligence** (DAI)
  - the field appeared around 1975-1980 with the goal of offering distributed (descentralized) solutions to AI problems.
  - the focus on of DAI is on getting agents to work together well to solve problems that require collective effort;
  - the agents formulate solutions by each solving (one or more) subproblems and synthesizing these sub-solutions into overall solutions.
- solving **distributed problems** well demands both:
  - group **coherence**
    - the agents need to want to work together
    - we assume that the agents have been designed to work together
      - ➢ we assume a fair degree of coherence is already present
  - group **competence**
    - agents need to know how to work together well
    - **DPS concentrates on competence**
- if the problem the agents are solving is to construct a **plan** (strategy, sequence of actions) $\Rightarrow$ **planning problems** (connected to *deliberative agents*).
  - planning problem
    - decompose problem in subproblems
    - allocate the subproblems
    - exchange problem solutions

        - synthesize overall solutions
    - if multiple agents are used for solving a *planning* problem ⇒ **distributed planning (DP)**
- **Characteristics of DPS**
    - speed up the solution process;
    - expertise and problem solving may be distributed;
    - the knowledge may be distributed;
    - the solution may need to be distributed.
- **Why DPS and DP?**
    a. using distributed resources concurrently leads to a speedup of the problem solving process (due to parallelism)
    b. expertise or problem solving capabilities can be distributed, e.g.
        - *concurrent engineering*
            - ➢ e.g. designing and manufacturing an artifact (car) by
                - ✓ allowing specialized agents to individually formulate components and processes
                - ✓ combining these into a collective solution
        - supervisory systems for *air-traffic control*
    c. data can be distributed
    d. the results of the problem solving or planning may need to be distributed

- **Example problems**
    - well known problems from the DAI field
        - ToH
        - Distributed Sensor Network Establishment (DSNE)
        - Distributed Vehicle Monitoring (DVM)
        - Distributed Delivery (DD)

DPS strategies

- ✓ **Task sharing**
    - main steps
        - ✓ **task decomposition**
            - decomposing large tasks into subtasks that could be handled by different agents
            - e.g. means-ends analysis
        - ✓ **task allocation**
            - assigns subtasks to appropriate agents
        - ✓ **task accomplishment**
            - the appropriate agents accomplish their subtasks
        - ✓ **result synthesis**
            - when an agent accomplishes its subtasks, it passes the result to the appropriate agent
    - task sharing in **homogeneous systems**
        - ✓ ToH problem
        - ✓ the agents have the same expertise
    - task sharing in **heterogeneous systems**
        - ✓ different capabilities for agents with different expertise
        - ✓ e.g. designing a car
- ✓ **Result sharing**

        - agents will separately and independently to derive different solutions ⇒ result sharing for improving group performance

Distributed planning (DP)

- is a specialization of DPS, where the problem to be solved is to construct a **plan (**strategy or sequence of actions)
    - e.g. autonomous robots, autonomous vehicles, games (*sequential environments*) – the MINIMAX algorithm is an example of a planning technique
- DP is an extension of classical **planning**
    - intensively researched subfield of AI, also connected to the field of (deliberative) *agents*
- **Planning**
    - deals with means to decompose a problem in subproblems and means to handle the interaction between subproblems while they are detected during the problem solving process
        - STRIPS (STanford Research Institute Problem Solver)
            - for robot control
            - idea in planning
                - we are given
                    - the description of the starting state
                    - goal state
                    - a set of possible actions

                  ⇒ find a sequence of actions from the start state to the goal state
    - is related to ***decision theory***
    - in known environment, planning can be done **offline (**solution can be found and evaluated prior to the execution)
    - in unknown environments, the strategy needs to be revised **online ⇒ trial-and-error**
        - e.g. reinforcement learning

3. An example of DPS. Distributed data mining

Using MAS for DPS

- the data is **distributed**
    - the classical perspective when an agent autonomously performs a (sub)task on the local data
        - stationary agents
        - mobile agents
    - descentralized approach
- the data is not spatially distributed **(**i.e. it is **centralized)**, but multiple agents are used for increasing performance (e.g. Vacuum cleaning with multiple robots)

3.1 Data mining (DM)

- **Data mining** (DM) or knowledge discovery in databases (KDD) is the process of search for meaningful information in large volumes of data.
    - discover patterns, rules.
    - extract knowledge from data

- DM is a multidisciplinary research topic in at least three domains:
    1. **Statistics**
        - linear regression, multiple regression and nonlinear regression are the main statistical methods used in DM
    2. **Artificial Intelligence.** The main methods come from Computational Intelligence (CI)
        - **ML** - learn relationships from data
            i. **Supervised**
            ii. **Unsupervised**
        - Fuzzy systems, evolutionary computation, soft computing, etc
    3. **Databases**
        - **Association rule (AR) mining**
        - E.g. **market basket analysis**

3.2 Distributed Data Mining (DDM)

- mining of distributed data sets.
    - the data sets are stored on local computers which are connected through a network
    - DM takes place at a **local level** (i.e. on a local data set) and on a **global level** where the local mining results are combined to obtain global findings (see Figure 1).
- also referred in the literature as **Distributed Learning**
- three major aspects related to DDM
    - algorithms for DDM (e.g. distributed clustering, distributed AR mining, distributed decision tree learning, etc)
    - **architectures and systems for DDM** (see Section 2.3)
    - applications of DDM to real world problems
        - e.g. intrusion detection, credit fraud detection, text classification, financial data mining from mobile devices, e-commerce, etc.

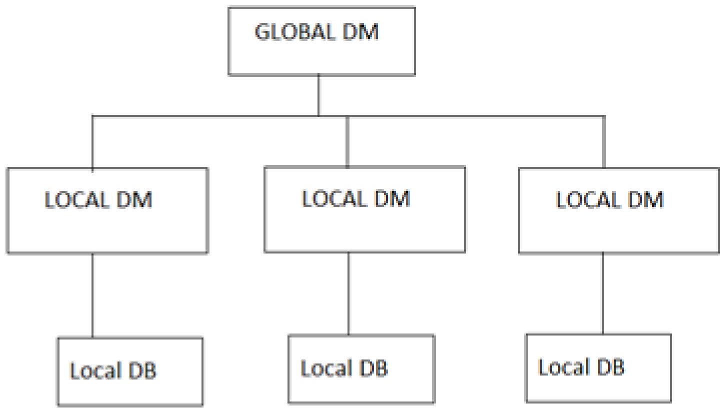


Figure 1. A DDM architecture

3.3 Architectures for DDM

- There are two main **models** for developing a DDM architecture
    1. the **client-server model**

        - one (or more) DM server collects data from different locations and is, generally, a high performance server
        - the users' requests are sent to the DM server which stores the mining algorithms
        - **drawback:** high communication overhead – the distributed data has to be collected at a central location
    2. the **agent-based model**
        - using **stationary agents** (i.e. are not mobile)
        - using **mobile agents**
            - from security reasons, when the user is mining highly sensitive data (e.g. medical) that should not leave the owner's site
            - in the mobile-agent model the mining algorithm and the relevant parameters are sent to the data site and at the end of the process the mobile agent is destroyed on the site and moves to another location where the execution restarts.

- In the agent-based model, each local database (data site) has one or more associated agents.

- The agents (**DM agents**) communicate the results to the other agents or to a central agent that supervises the system (**supervisor agent**) – see Figure 2.
    - local agents correspond to local DBs
    - a central agent corresponding to the central DB
    - each agent mines data from its local DB and communicates the result to the central agent
    - the results of the local mining are combined by the central agent
- DM agents can work in parallel and share information they have gathered
- **<u>Main advantage of using agents in DM</u>** – possible support for **online/active data mining**
    - when new data is added to the local DB, an alarm/triggering agent may notify the main mining application so that the new data can be evaluated with the already mined data
    - this is called **active data mining**

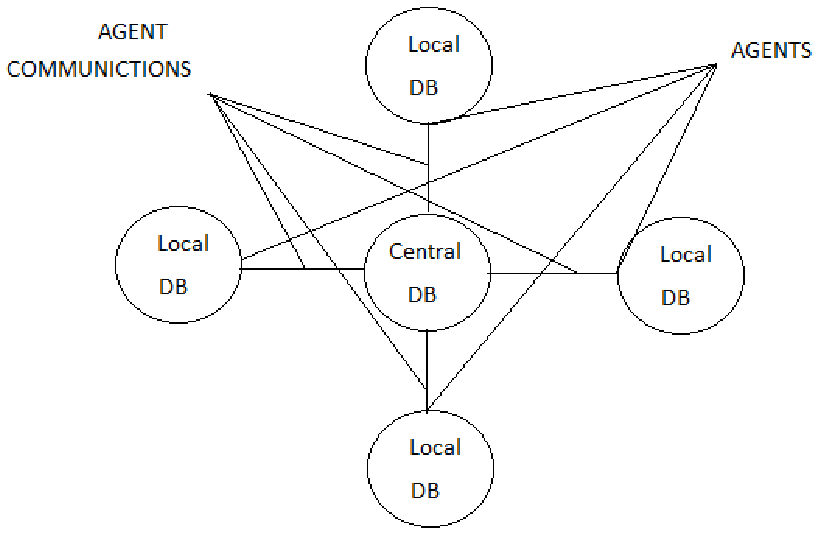


Figure 2. Architecture of a DDM system using agents

Bibliography

[1] Weiss, G. (Ed.): Multiagent Systems: *A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999 (available at www.cs.ubbcluj.ro/~gabis/weiss/Weiss.zip) [Ch. 2]

[2] Șerban Gabriela, *Sisteme Multiagent în Inteligenta Artificiala Distribuita. Arhitecturi si Aplicatii*, Editura RisoPrint, Cluj-Napoca, 2006

[3] Czibula Gabriela, *Sisteme inteligente. Instruire automată*, Editura RisoPrint, Cluj-Napoca, 2008