Introduction

Distributed Artificial Intelligence (DAI) and Multiagent Systems (MAS) share a lot of common ground with distributed systems. Interesting problems in AI are being formulated in terms of spatially, functionally or temporally distributed processing.

**DPS** is a subdiscipline of AI concerned with **distributing and coordinating** knowledge in multiple agent environments.

**MAS** is concerned with the behavior of **autonomous** agents aiming at solving a given problem.
DAI vs MAS

- Distinction between DAI and MAS became blurred
  - DAI focus was on how to do Distributed Problem Solving (DPS)
  - DPS took for granted that agents would be able to agree, share tasks, communicate truthfully, and so on.
  - In other words, DAI ignored social aspects of agency
  - MAS borrowed the underlying assumption from game theory and social science that an agent should be rational. It should endeavor to maximize its own benefit/payout.
  - MAS has, for all practical purposes, replaced DAI

- DAI focus’ is on the interaction between agents situated in a common environment.
  - Two subfields:
    - DPS which is strictly cooperative
    - MAS which is focused on coordination among self-interested agents.

Enabling Concepts and Technologies

**Ubiquity**
- Continual reduction in cost of computing capability
- What can benefit from having an embedded processor in it?

**Intelligence**
- The complexity of tasks that we are capable of automating and delegating to computers has grown steadily

**Human orientation**
- Movement away from machine-oriented views of programming toward concepts that reflect the way we understand the world

**Global computing**
- Novel software development models are needed to deal with it

**Interconnection and Distribution**
- Core motifs in Computer Science
- Implies systems that can cooperate and reach agreements
What is an agent?

• Building block for a **MAS**
• No universally agreed definition
  - Agent abstraction encapsulates an adaptability and flexibility

  “Agents are simply computer systems that are capable of autonomous action in some environment in order to meet their design objectives. An agent will typically sense its environment by physical sensors, in the case of agents situated in the real world, or by software sensors in the case of software agents.”  **Wooldridge**

Types of Agents

• **Simple reflex agents**: respond directly to percepts.
• **Model-based reflex agents**: maintain an internal state to track aspects of the world (environment) that are not evident in the current percept.
• **Goal-based agents**: act to achieve their goals
• **Utility-based agents**: try to maximize their own expected happiness.

This list can be further subdivided into: table driven agents, problem solving agents, deliberative agents, decision-theoretic or rational agents, knowledge based agents, planning agents, reactive agents, proactive agents, adaptive agents and learning agents.

• All agents can improve their performance by **learning**
Intelligent Agents

- **Autonomous**: capable of independent action
- **Situated**: embedded in an environment
- **Proactive**: responsive to changes in the environment
- **Rational**: goal-oriented, striving to optimize its decisions
- **Socially capable**: able to communicate and interact with others

**Intelligent agents** are the most sophisticated kind of agent

Environment

- Extremely important part of a MAS
- Includes everything that was not contained initially in the agent, including other agents
- Characterization [Russell & Norvig]
  - Accessible vs inaccessible
    - Can the agent access all the environment information?
  - Deterministic vs non-deterministic
    - Is there no uncertainty about the effect of an action?
  - Episodic vs non-episodic
    - Are there discrete episodes in the environment?
  - Static vs dynamic
    - Is the environment changing or not?
  - Discrete vs continuous
    - Are there a fixed, finite number of actions?

“Agents and their environments are inextricably coupled! Simple agents can often display apparently complex behaviors when they interact with sufficiently rich environments!” **Honavar**
### Agent Architectures

- Deals with the design and implementation of computer systems that satisfy properties specified by agent theorists
  - A particular methodology for building agents
  - Decomposition into a set of component modules
  - How these modules interact
  - Encompasses techniques and algorithms

- Three basic types:
  - **Deliberative**: Explicitly represented, symbolic model of the world. Decisions about actions to perform are made via logical reasoning based on pattern matching and symbolic manipulation.
  - **Reactive**: No central symbolic world model, no complex symbolic reasoning.
  - **Hybrid**: Has both deliberative and reactive components.

### What is a MAS?

- More than one agent in a given environment
  - A number of agents, which interact with one another.
  - In the general case, agents will be acting on behalf of users with different goals and motivations.

- To interact successfully, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do.

- Characteristics of MAS [Sycara]
  - Each agent has incomplete information or capabilities for solving the problem and thus has a limited viewpoint
  - There is no system global control
  - Data is decentralized
  - Computation is asynchronous
Why use MAS?

Are not centralized solutions more efficient? **Not always!**

1. Problem being solved is itself distributed.
2. Distribution can lead to algorithms that might not have been discovered with a centralized approach.
3. A centralized approach is impossible.
4. The information involved is necessarily distributed, and resides in information systems that are large and complex in several senses.
5. Components of systems are typically distributed and heterogeneous. [Huhns]

---

Why use MAS?

1. To solve problems that are too large for a centralized agent to solve.
2. To allow for the interconnection and interoperation of multiple existing legacy systems.
3. To provide solutions to problems that can naturally be regarded as a society of autonomous interacting agents.
4. To provide solutions that efficiently use information sources that are spatially distributed. E.g., sensor networks, seismic monitoring, and information gathering from the internet.
5. To provide solutions in situations where expertise is distributed.
6. To enhance performance. [Sycara]
Distributed Systems and MAS

• “A distributed system is a collection of independent computers that appears to its users as a single coherent system.” [Tanenbaum]

• It is apparent from the previous discussion that MAS are distributed systems.

• Distributed systems deal with architectures, processes, communication, synchronization, consistency and replication, and fault tolerance.

• MAS deals with coordination (cooperation and negotiation), communication, dependability, fault tolerance, computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reuse.

Distributed Systems and MAS

• Major difference between distributed systems and MAS
  - Distributed Systems: desired behavior is planned at design time.
    Allows for little or no deviation.
  - MAS: designed such that their behavior is flexible and adaptable to the ever changing environment.

• For MAS, the behavior of a system, as a whole, is emergent.

• It arises out of the interactions between all the agents in the system and their environment.

• This requires coordination, cooperation and communication.
Coordination

- **Coordination** is a key problem in DAI/MAS.

  Coordination is the process through which agents reason about their local actions, anticipate the action of other agents and try to ensure the community acts in a coherent manner.

  - There are three main reasons why the actions of multiple agents need to be coordinated:
    1. There are dependencies between agents' actions.
    2. There is a need to meet global constraints.
    3. No one individual has sufficient competence, resources or information to solve the entire problem.

Coordination

- Ideally all agents in a system would have complete knowledge of the goals, actions and interactions of their fellow agents. Such a perfectly coordinated system is impossible.

  - The next best possible solution would be to have a global agent controller. This global agent controller could direct the activities of the others, assign agents to tasks and focus problem solving to ensure coherent behavior. This approach is impractical and would become a performance bottleneck and a single point of failure.

  - Most research has concentrated on developing agent communities in which both control and data are distributed.
Coordination

- **Distributed control** means that individuals have a degree of autonomy in generating new actions and in deciding which tasks to do next.

- **Important Consideration**: When designing such systems it is important to ensure that agents spend the bulk of their time *engaged on solving the domain level problems* for which they were built, *rather than in communication and coordination activities*.

- Coordination depends on whether agents are *cooperative* or not.

Cooperative vs Non-Cooperative Agent Systems

- **Cooperative agents systems**: A group of *cooperative* agents works jointly to achieve a common goal.

- **Non-cooperative agent systems**: A group of *self interested* agents interacting in a shared environment. -Agents can either be *adversarial* or *indifferent*. 
Cooperative vs Non-Cooperative Agent Systems

• Key research issues for **cooperative agents:**

1. How do agents decompose the goal into subgoals that can be assigned to individual agents based on their capabilities and access to resources?
2. How to develop agent organizations (authority relationships) and problem-solving protocols (information flow) that enable agents to share results and knowledge in a timely, effective manner?
3. How do agents maintain coherence and problem-solving focus when locally available information can be incorrect, inconsistent, outdated, etc.?

Cooperative vs Non-Cooperative Agent Systems

• Key research issues for **non-cooperative agents adversarial agents:**

1. Modeling the knowledge and behavioral strategies of opponents.
2. Learning to exploit opponent weakness.
3. Developing interaction rules by which agents can arrive at equilibrium configurations.

• Key research issues for **non-cooperative agents indifferent agents:**

1. Designing social laws, conventions, and protocols by which each agent can achieve its own goal without significantly affecting the chances of others achieving their goals.
Communication Protocols

- **Communication protocols** enable agents to exchange and understand messages.
- **Interaction protocols** enable agents to have conversations, defined as structured exchange of messages.

- A communication protocol might specify the following type of messages that can be exchanged between two agents:
  - Propose a course of action
  - Accept a course of action
  - Reject a course of action
  - Retract a course of action
  - Disagree with a proposed course of action
  - Counterpropose a course of action

Communication Protocols

- **Communication** can enable the agents to coordinate their actions and behavior resulting in systems that are more **coherent**.
- It is important for agents of **different capabilities** to be able to communicate. Communication must defined at several levels, with communication at the lowest level used for communications with the **least capable agent**.
- Spoken human communication is used as a model for communication between computational agents. Speech act theory views human natural language as actions, such as **requests**, **suggestions**, **commitments** and **replies**.
Communication Protocols

- A number of protocols and languages have been defined for agent communication. Examples are the Knowledge Query and Manipulation Language (KQLM), a protocol for exchanging information and knowledge and the Knowledge Interchange Format (KIF), a computer-oriented language for the interchange of knowledge among disparate computer programs.

Implementation Issues

- DAI and MAS need to address a number of implementation issues that in themselves are complex challenges.
  - Task decomposition and allocation
  - Agent communication and interaction
  - Coordination and coherence (cooperation/negotiation)
  - Reason about the actions, plans, and knowledge of other agents
  - Conflict resolution
  - Practical DAI/MAS system design including protocols design
**Analysis**

- Replacement of DAI by MAS should not be surprising
- DAI can be seen as a reductionist approach.
  - System is divided into modules that can be solved by the application of the appropriate algorithms.
  - Knowledge is designed into the system, contained into its algorithms.
  - DAI and DPS are cooperative endeavors where a key assumption is that all agents are working together.
- MAS, has an emergent approach
  - A good amount of design and algorithms go into creating a MAS, but it is the behavior that emerges when the system is operating, that gives MAS their power.
  - Embraced the social aspects of agency that DAI ignored.
  - Makes **no assumptions** about other agents. Agents can be cooperative, adversarial or indifferent.
  - Key consideration is to create flexible and adaptable systems.

**Speculation**

- One problem with agents and MAS is that the **burden for designing and implementing one is very high**.
- Agent oriented programming can be considered to be in its infancy.
- Developers still have to implement the agents, and their corresponding coordination protocols and communication protocols from scratch using very low level of abstraction to create higher level abstractions.
- Agents and MAS are as high level abstractions as they come.
- A **possible remedy** for this is to create a programming language that while supporting the usual high level language constructs (iteration, decision) it supports the creation of agents as well as supporting infrastructure and architectures for MAS.