Advanced Robotics
Multi-Agent Systems/Communication

Gerald Steinbauer
Institute for Software Technology
Agenda

• Motivation
• Example Domain
• Definitions and Properties of Multi-Agent Systems (MAS)
• Task Allocation
• Communication in MAS
Motivation

• **many** can maybe achieve what one can not
• the way humans **solve** some problems
• multi-agent system means **more** than a group of robots
• motivated by “**distributed AI**”
• interoperability of **heterogeneous** systems
• growing **interest**
  • computational resources can be easily **distributed**
  • even small entities have computational and communication capabilities
• very **active** research field (e.g. AAMAS conference)
Examples for Multi-Agent Systems
Agent Competition

- large scale disaster simulation
- simulators for earthquake, fire, civilians, and traffic
- the task is to develop software agents with different roles, that
  - make roads passable (police)
  - extinguish the fires (fire brigades)
  - rescue all civilians (ambulances)
- a challenging Multi-Agent Problem since Agents must cooperate
- the more civilians alive and fires put out, the higher the score
RoboCup Rescue Scenario

example application: earthquake in a given city
Foligno – dense building layout

- Fire Brigade
- Police
- Ambulance (Team)
- Unhurt Civilian
- Hurt Civilian
- Dead Civilian
- Road Block
Platoon Agents

**Fire Brigades**
- extinguish fires
- have limited water tank capacity
- can collaborate to extinguish more quickly

**Ambulances**
- dig out civilians who are buried under rubble (takes a number of cycles)
- carry victims to a refuge
- can collaborate to dig out more quickly

**Police Force**
- can remove obstructions on roads
- cannot collaborate to remove obstructions faster
Centre Agents

- **Ambulance Centre**
  - talks to ambulances and to other centres
- **Fire Centre**
  - talks to fire brigades and other centres
- **Police Centre**
  - talks to police force agents and other centres

- located in a **building** which may or may not collapse or get burnt
- usually act as a **synchronisation point/computation centre**
Kobe Map

long distances, loose building layout, sparse bits
Random Maps

with different building/road densities
Exploration

https://www.youtube.com/watch?v=zw8DlpCsjBc

Attacking Fires
Challenges

- heterogeneous agents
- a large fixed number of agents
- a variable number of tasks
- dynamic environment
- incomplete knowledge
- limited communication
- touches a number of research areas
  - team exploration (victim search)
  - multi-agent path planning (removal of road blockage)
  - scheduling (treatment of victims)
Agent

- there is no generally agreed definition
- there is a good one by Wooldridge

“an agent is a computer system that is situated in some environment and is capable of independent (autonomous) action in this environment on behalf of its user or owner in order to meet its design objectives.”

- related to controller, reactive systems, robots, …
Properties of an Intelligent Agent

• **autonomous**
  • it has control of its own **actions**
  • it can make its own **decision**

• **reactive**
  • it reacts to events or changes in its **environment**
  • it **response** in time

• **proactive**
  • it acts on its own **initiative**
  • act to achieve its **goals**

• **social**
  • it **interacts** with other agents
  • it **collaborate** or negotiate with others
Rational Agent

• a definition more related to AI and robotics

“for each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measures, given the evidence provided by the percept sequence and whatever build-in knowledge the agent has.”
Multi-Agent System (MAS)

“a MAS can be defined as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capability or knowledge of each problem solver”

• these problem solvers, called agents, are autonomous and can be heterogeneous in nature

• to successfully interact, the agent will require the ability to communicate, cooperate, coordinate and negotiate like humans do
Social Ability

- **communicate**: the ability to exchange message and information
- **cooperate**: working together as a team to achieve a shared goal (not able to achieve alone, better results)
- **coordinate**: managing the interdependencies between activities (non-shareable resources)
- **negotiate**: the ability to reach agreements on matters of common interest (use of TV set)
Assignment Problems

- assigning $n$ agents to $m$ tasks
  - for example, passengers to drivers (car sharing domain)
  - cleaning robots to rooms
  - wedding matching with individual sympathy measure
  - ...

- individual assignment costs $c_{ij}$ occur for each assignment of agent $i$ to task $j$

- the objective is to find a one-to-one matching from agents to tasks at the least possible total cost
Task Allocation

• a (basic) task allocation problem is given by
  1. a set of tasks $T, < t_1, \ldots, t_n >$
  2. a set of robots $R, < r_1, \ldots, r_m >$
  3. a assignment matrix $A$, $a_{ij} = 1$ if task $i$ is assigned to robot $j$ otherwise 0
  4. a utility function $U: A \rightarrow \mathbb{R}$ defining the value or reward of executing tasks by particular robots
  5. maximize the reward $\text{argmax}_A U$
Solutions to the TA Problem

- **communication**
  - the approaches differ in their nature
    - centralized: needs $n^2$ messages to distribute all utilities, works fine with $n < 200$
    - distributed: needs fewer messages, i.e., less than $n$
  - in some larger systems communication **latency** cannot be ignored

- **no communication**
  - estimate the **utility**
  - estimate the **intentions** and **actions** of the other agents
Optimal Assignment Problem

find $m \times n$ non-negative integers $x_{ij}$ to maximize

$$U = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} U_{ij} \omega_j$$

$$\sum_{i=1}^{m} x_{ij} = 1 \leq 1 \quad 1 \leq j \leq n \quad i^{th} \text{ agent will do one job}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \leq 1 \quad 1 \leq i \leq n \quad j^{th} \text{ job will be done by only one agent}$$

$x_{ij} \in \{0,1\}$

$n \neq m$

$\omega_j$ are the weights for the prioritization of tasks
Heterogeneous Agents

• agents have a qualification for particular jobs

$$\text{maximize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} U_{ij}$$

subject to:

$$\sum_{i=1}^{m} q_{ij} x_{ij} \leq 1 \quad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} q_{ij} x_{ij} \leq 1 \quad i = 1, \ldots, m$$

$$x_{ij} \in \{0,1\}$$

• where $x_{ij} = 1$ agent $i$ is assigned to task $j$ and

• $q_{ij} = 1$ if agent $i$ is qualified for task $j$
Linear Programming

- OAP can be formalized as **linear program**

- A linear program **comprises**:
  - A **vector** of non-negative variables $x, x_i \geq 0$
  - A linear **objective** function to be maximized $c^T x$
  - Constraints equations $Ax \leq b$
  - Non-negative right hand side constant $b_i \geq 0$

- Algorithms to find the **optimal** solution
  - Simplex
  - Hungarian Method $O(mn^2)$

- Or use a state-of-the-art **LP solver**, e.g. minizinc, CPLEX
Hungarian Method I

- **Step 1**: subtract the **smallest** entry in each **row** from all the entries of its row
- **Step 2**: subtract the **smallest** entry in each **column** from all the entries of its column
- **Step 3**: draw **lines** through appropriate rows and columns so that all the zero entries of the cost matrix are **covered** and the **minimum** number of such lines is used
Hungarian Method II

• **Step 4:** Test for Optimality
  - (i) if the minimum number of covering lines is \( n \), an optimal assignment of zeros is possible and we are finished
  - (ii) if the minimum number of covering lines is less than \( n \), an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5.

• **Step 5:** determine the smallest entry not covered by any line. subtract this entry from each uncovered row, and then add it to each covered column. return to Step 3.
Auction Algorithms I

• tasks are sold to agents by a broker during an auction

• for optimal auction algorithms
  • construct a price-based “task market”
  • the broker assign a value $c_j$ on each task $j$
  • each robot $i$ also assign a value $a_{ij}$ on task $j$
  • in order to be sold by the broker there must be a price $p_j$ greater as $c_j$

• a robot is elect to buy the task with the most profit
  \[ t_i = \max_j \{a_{ij} - p_j\} \]

• a robot is not “happy” if the above condition is not satisfied
Auction Algorithm II

• the auction mechanism is conducted in rounds and works as follows:

1. start with any assignment and any prices for \(i\)
2. if all robots are “happy” then terminate otherwise select an “unhappy “ robot \(i\) and find the best task \(j_i = \arg\max_j\{a_{ij} - p_j\}\)
3. exchange the task with the robot assigned before
4. set the price of task \(j_i\) so that is indifferent to the second best task: \(p_j + \gamma_i, \gamma_i = \nu_i - w_i, \nu_i = \max_j\{a_{ij} - p_j\}, w_i = \max_{j \neq j_i}\{a_{ij} - p_j\}\)
5. repeat until all robots are “happy”
Auction Algorithm III

- **problem**: the algorithm may not terminate if more than one object offers maximum value to robot $i$, $\gamma_i = 0$
- to break the cycle $\gamma_i$ has to be a positive increment
- a robot is almost "happy" with task $j_i$ if
  \[ a_{ij_i} - p_{j_i} \geq \max_j \{a_{ij} - p_j\} - \varepsilon \]
- if we chose $\gamma_i = v_i - w_i + \varepsilon$ robot $j$ is almost happy after each round
- this guarantees that the process terminates after a finite number of rounds ($n$ rounds) and it reaches almost the equilibrium
- if all $a_{ij}$ are integer and $\varepsilon < 1/n$ a reached assignment is optimal
Agent Communication

- **communication in concurrent processes**
  - synchronization of multiple processes
  - access to shared resources

- **communication in OOP**
  - method invocation between different modules
  - object $o2$ invokes method $m1$ on object $o1$, arguments and results to be communicated

- **communication in MAS**
  - autonomous agents have control over both state and behavior
  - methods are executed according the agent’s self-interest
  - agents can perform communication to influence other agents
  - agent communication implies interaction
  - is a central part of MAS
Speech Act

- most treatment of communication is MAS is inspired by speech act theory
- the theory of speech acts is generally recognized to have begun with the work of the philosopher John Austin
- speech act theory studied the pragmatic use of language
  - an attempt to account for how language is used by people every day to achieve their goals and intentions
- speech act theory treats communication as action
  - speech actions are performed by agents just like other actions
Speech Act

- Austin noticed that some utterance are rather like “physical actions” that appear to change the state of the world
  - pragmatic examples:
    - declaring war
    - “I now pronounce you man and wife”
  - but more generally, everything we utter is uttered with the intention of satisfying some goal or intention
  - Austin identified a number of performative verbs, which correspond to various different types of speech acts
    - examples are request, inform and promise
Speech Act

• Searle (1969) identified the following five key classes of possible types of speech acts:
  • representatives: commits the speaker to the truth of an expression, e.g., “it is raining” (informing)
  • directives: attempts to get the hearer to do something, e.g., “please make the tea” (requesting)
  • commissives: which commits the to do something, e.g., “I promise to …” (promising)
  • expressives: whereby a speaker expresses a mental state, e.g., “thank you” (thanking)
  • declaration: effect change of state, such as “declaring war” (declaring)

• Cohen and Parrault (1979) modeled speech acts in a planning system (STRIPS notation)
Agent Communication Languages

- communication is a **central** issue in MAS
- agent communication languages (ACL) are **standard** formats for the **exchanges** of messages
- **message**: typically contains the following components
  - a **performative** such as inform, request, or accept
  - the actual **content** of the massage (application dependent)
  - name of **sender** and **receiver**, probably a timestamp
- **semantics**: try to explain the meaning of ACLs
  - Ann may send an *inform*(X) message to Bob only if she **believes** X to be true
  - Bob may send a *request*(Y) message to Ann only if he believes that Ann does not already intend to perform action Y
Agent Communication Languages

- the Knowledge Sharing Effort (KSE) of the ARPA in early 1990s designed two ACLs with different purposes
- agreed standards
- the Knowledge Query and Manipulation Language (KQML), which defines performatives and message formats
- the Knowledge Interchange Format (KIF), a language for expressing content, closely based on first order logic
Knowledge Interchange Format (KIF)

- KIF allows agent to express
  - properties of things in a domain, e.g., “Michael is an vegetarian”
  - relations between things in a domain, e.g., “Michael and Jane are married”
  - general properties of a domain, e.g., “all students are registered for at least one course” (quantification)

- examples
  - “the temperature of m1 is 83 degree Celsius”: (= (temperature m1) (scalar 83 Celsius))
  - “an object is a bachelor if the object is a man and is not married”: (defrelation bachelor (?x):= (and (man ?x) (not (married ?x))))
  - “any object with property being a person also as the property of being a mammal”: (defrelation person (?x): => (mammal ?x))
Knowledge Query and Manipulation Language (KQML)

• KQML defines communication verbs, or performatives, for example:
  • ask-if (“is it true that …”)
  • perform (“please perform the following action …”)
  • tell (“it is true that …”)
  • reply (“the answer is …”)

• each message has a performative (the “class” of a message)

```
(ask-one
 :content (PRICE IBM ?PRICE)
 :receiver stockServer
 :language LPROLOG
 :ontology NYSE-TICKS)
```
# Knowledge Query and Manipulation Language (KQML)

- **parameter** of the message
  - **:content**: content of the message
  - **:language**: formal language the message is in
  - **:ontology**: terminology the message is based on
  - **:force**: will sender ever deny content of message?
  - **:reply-with**: reply expected? identifier of reply?
  - **:in-reply-to**: id of reply
  - **:sender**: sender
  - **:receiver**: receiver
(evaluate
  :sender A :receiver B
  :language KIF :ontology motors
  :reply-with q1 :content (val (torque m1)))

(reply
  :sender B :receiver A
  :language KIF :ontology motors
  :in-reply-to q1 :content (= (torque m1) (scalar 12 kgf)))

(stream-about
  :sender A :receiver B
  :language KIF :ontology motors
  :reply-with q1 :content m1)

(tell
  :sender B :receiver A
  :in-reply-to q1 :content (= (torque m1) (scalar 12 kgf)))

(tell
  :sender B :receiver A
  :in-reply-to q1 :content (= (status m1) normal))

(eos
  :sender B :receiver A
  :in-reply-to q1)
Problems with KQML

- the basic KQML performatives set was overly large and **not standardized**
  - different implementations of KQML could not interoperate
- the language was **missing** the performative commissives
  - commissives are crucial for agents coordinating their tasks
- these problems – among others – lead to the development of **new** language by the FIPA consortium
FIPA (Foundation for Intelligent Physical Agents)

- FIPA is the **organization** for developing standards in multi-agent systems, accepted standard committee by IEEE
- FIPA’s goal in creating agent standards is to promote **interoperable** agent applications and agent systems
- FIPA ACL’s syntax and basic concepts are very similar to KQML
  - **performatives**: 20 performatives in FIPA
  - **housekeeping**: e.g., sender etc.
  - **content**: the actual content of the message
# FIPA ACL

<table>
<thead>
<tr>
<th>performative</th>
<th>passing info</th>
<th>requesting info</th>
<th>negotiation</th>
<th>performing actions</th>
<th>error handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept-proposal</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cancel</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>confirm</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>disconfirm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inform</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inform-if</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inform-ref</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not-understood</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>propose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>query-if</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query-ref</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>refuse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reject-proposal</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>request-when</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>request-whenever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subscribe</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FIPA - Syntax

(inform
  :sender agent1
  :receiver agent2
  :content (price good2 150)
  :language sl
  :ontology hpl-auction
)

FIPA Interaction Protocols (IPs)

• Interaction Protocols (IPs) are standardized exchanges of performatives according to well known situations

  • FIPAResquest
  • FIPAPquery
  • FIPAResquestWhen
  • FIPACcontractNet
  • FIPAIteratedContractNet

• FIPAAuctionEnglish
• FIPAAuctionDutch
• FIPABrokering
• FIPARecruiting
• FIPASubscribe
• FIPAPPropose
FIPA Interaction Protocols (IPs) - Request

- Request
- Refuse
- Agree
- Failure
- Inform-Done
- Inform-Results
Contract Net

• well known task-sharing protocol for task allocation is the contract net

• the contract net includes five stages:
  1. recognition
  2. announcement
  3. bidding
  4. awarding
  5. expediting

• example: delivery task
Contract Net - Example

© Alexander Kleiner
Ontology

• ontologies grounds the terminology used by the agents
  • for example, an agent wants to buy a screw. But what means then size? It is in inch or centimeter?

• very important in the Internet, sometimes encoded by XML
  • in contrast to HTML whose meta-language mainly describes the page layout, XML allows to tag data with semantics → semantic web
Java Agent Development Framework - JADE

- open source project by Telecom Italia (TILAB), governed by an international board
- it allows the rapid creation of distributed multi-agent systems in Java
- high interoperability through FIPA compliance
JADE offers

• a library for developing agents; which implements message transport and parsing
• a runtime environment allowing multiple, parallel and concurrent agent activities
• graphical tools that support monitoring, logging and debugging
• yellow pages, a directory where agent can register their capabilities and search for other agents and services
JADE – System Overview

Multi-agent application

Homogeneous layer

Platform

Java

Wireless and wireline

Internet
JADE - Tools
RoboCup Logistics League (RLL)

- competition of groups of robots
- efficiently manufacture a product using different steps and machines
- coordination, planning and execution in a MAS
- http://www.robocup-logistics.org/
- we are looking for students … (project, theses, exercises)

https://www.youtube.com/watch?v=_iesqH6bNsY

https://www.youtube.com/watch?t=11&v=fT4S2kuLMAo
RLL and YAGI

• together with RWTH Aachen and the University of Rome we used YAGI for high-level control in the RLL
Literature


Thank you!