

Multiagent Systems and Agent-based Modeling and Simulation

Ana L. C. Bazzan
Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS, Brazil

PPSN 2018 – Sept. 08

About me...

- PhD University of Karlsruhe, Germany
- doing Multiagent Systems (MAS): since 1993
- member of IFAAMAS board (2004–2008, 2014–current)
- chair AAMAS 2014 conf. in Paris
- plenary speaker AAMAS 2017
- assoc. editor JAAMAS (Kluwer/Springer) from 2010 to 2017
- assoc. editor Adv. in Complex Systems from 2007 to 2017
- organization of several workshops and meetings around MAS, machine learning and complex systems

What my group does...

- Approaches
 - Multiagent reinforcement learning
 - Coordination in MAS
 - Game theoretic decision making
 - Self organization using swarm intelligence
 - Agent-based modeling and simulation (ABMS)
 - Complex systems; networks
- Applications
 - Traffic: control and routing
 - Task allocation in MAS
 - Smart cities
 - History and teledramaturgy

What you can expect from this tutorial

- Traditional AI versus distributed AI
- Overview on multiagent AI: autonomous agents and multiagent systems
- Discussion about which problems arise when a system has more than one interacting agent
- Overview on agent-based modeling and simulation (ABMS)
- Cross-fertilization: evolutionary algorithms and multiagent systems
- Review on works in the frontier
- Open challenges

Recommended Reading

- Shoham and Leyton-Brown: Multiagent Systems
- M. Wooldridge: Multiagent Systems
- Nikos Vlassis: A Concise Introduction to Multiagent Systems and Distributed AI
- Stone, P. and Veloso, M. : Multiagent systems: a survey from a machine learning perspective. Autonomous Robots, 8(3), 2000
- Leyton-Brown and Shoham: Essentials of Game Theory
- Articles in conferences and workshops: AAMAS, IJCAI, AAAI, ICML, EcoMAS, ALA, OptMAS

Outline

- 1 Overview
- 2 Introduction to AA & MAS
 - The Basics of AA and MAS
 - Monoagent vs. Multiagent AI
 - Multiagent Systems
- 3 Agent-based Modeling and Simulation
- 4 Planning and MDP
 - Sequential Planning and Decision Making for One Agent: MDP
 - Sequential Decision Making in MASs: MMDP / SG
- 5 Reinforcement Learning
 - Basics
 - Single-agent RL
 - Multiagent RL
- 6 Distributed Constraint Optimization
 - CSP and COP
 - DisCSP and DCOP
- 7 Integration ML and MAS
 - Agent-Based ML
 - ML in MASs
- 8 Challenges
- 9 References

Outline and Goals

- Kernel: show differences between single and multi agent AI
- Also: agent-based modeling and simulation
- 3 main areas of AI are (re)visited here (those closer to PPSN's topics)
 - Planning and decision making (mostly Markov decision processes)
 - Learning (mostly reinforcement learning)
 - Problem solving via constraint satisfaction

Why AA and MAS ?

- Decentralization: “the” rule in Nature
 - no leader in flocks of birds
 - no leader in colonies of ants (queen just lay eggs !)
 - ...
 - why people always think that a pattern exists only if someone creates or orchestrates it ???

Mono vs. Multiagent

● Example Figure Skating

one agent	multiagent	not as planned !
	  	

Lessons Learned

- One agent is easy (well...)
- Two agents: coordination !!!
- But: not all situations can be predicted...
- Therefore adaptation is key

What's an Agent?

- No standard, accepted definition
- Def. 1:

An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future. (Franklin and Graesser)

What's an Agent?

- Def. 2:

An agent is a computer system that is situated in some environment and that is capable of independent (autonomous) action in this environment on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told) (Wooldridge)

What's an Agent?

- (other) important concepts:
 - Situated
 - On behalf of its owner
 - Rather than constantly being told
 - Should be able to:
 - 1 Perceive the environment
 - 2 Decide which action to make (deliberation)
 - 3 Act in the environment

Agents and MAS

- Fulfills all 3 criteria?
 - It is an agent !
- Interacts with other agents?
 - It is a MAS
- Improves performance over time?
 - It is a learning agent

In Short

- Agents are
 - Autonomous
 - Flexible
 - Have own control
 - Interact (with other autonomous agents)
 - Are situated (in an **environment**)

Environment

- Environment = sensing (perception) + action
- Types of environment:
 - Totally vs. partially observable
 - Deterministic vs. non-deterministic
 - Episodic vs. non-episodic (sequential)
 - Discrete vs. continuous
 - Static vs. dynamic
 - Single agent vs. multiagent

Monoagent AI

- Revisiting some basics of (mono or single agent) AI
- Relevant areas to discuss role of multiagent
 - Planning and decision making
 - Learning
 - Problem solving via constraints satisfaction/optimization

Monoagent AI – Planning and decision making

- Planning task: find a sequence of actions to reach a given goal
- How to go from Berlin to Coimbra to present a tutorial?
 - agent that plans: look for routes, timetables, etc. and plans
 - assumes an initial state and a goal
 - another way is...

Monoagent AI – Planning and decision making

- Agent as utility maximizer
(via MDP - Markov decision processes)
 - Set of states S
 - Set of actions A
 - transitions
 - $S \times A \rightarrow \mathbb{R}$
- Able to deal with uncertainty

Monoagent AI – Planning and decision making

- Agent as utility maximizer
 - In principle:
 - possible to model any decision process of the agent via an MDP
 - In practice:
 - Some limitations (see later)

Monoagent AI – Planning and decision making

- This works fine for ONE agent...
- ... but: how about more than one?

Monoagent AI – Learning

- Learning
 - Important role in any system that intends to be intelligent
 - Set of algorithms that increase the ability of the agent to match a series of inputs to their corresponding outputs
- Important here:
 - Reinforcement learning (RL)
 - We do not tell the agent explicitly what to do or which action to make
 - Agent has to select the action that achieves the highest reward (initially by trial and error)

Monoagent AI – Learning

- This works fine for ONE agent...
- ... but: how about more than one?

Monoagent AI – Problem solving with CSP

- One approach to problem solving
 - Constraint satisfaction problem (CSP)
 - Set of variables
 - Set of constraints
 - Solution: assignment of values for each variable so that no constraint is violated

Monoagent AI – Problem solving with CSP

- CSPs may have no solution or have several solutions
- Thus: constraint optimization problem (COP)
 - Find the best solution
 - Quality criteria to evaluate each solution
 - Cost function

Monoagent AI – Problem solving with CSP

- This works fine for ONE agent...
- ... but: how about more than one?

Multiagent AI

- One vs. several agents:
 - Partial, local knowledge about the problem
 - ...
- How efficient are classical techniques?
- After all.... what changes?
- We will see this in details by revisiting each of those 3 areas

Definitions

- Definition of MAS: again, no definitive definition
- Def. 1:
*a multiagent system is a system composed by several agents that **interact***
- Def. 2:
*A multiagent system is one that consists of a number of agents, which **interact** with one another. To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do. (Wooldridge)*
- Multiagent systems and collective intelligence are **not** the same

Collective Intelligence



Collective Intelligence

- 1906 / UK: F. Galton visits an agricultural exposition
- Hundreds of “non experts” bet about the weight of an ox (which participates in the exposition)
- Galton believes that only experts (e.g., breeders) can correctly evaluate the weight
- But: the average weight comes out correctly !!!

Collective Intelligence

- 1987 / Challenger explosion:
- Financial market “knew”, half an hour after the accident, that the Thiokol company was responsible (Thiokol shares went down 12% that day, while shares of other suppliers went down only 3%)
- This did not appear in the press
- Technical committee in charge: report after 6 months charging Thiokol

Collective Intelligence

- Iowa Electronic Markets (IEM): forecast elections results
 - 1998–2000: error of just 1.37% in the Bush vs. Al Gore presidential election; 3.43% in other American elections; 2.12% foreigner elections
 - Performance superior to opinion polls
 - Hollywood Stock Exchange (HSX): box office, Oscar, etc.

Collective Intelligence

- 4 conditions for a wise collective:
 - ➊ diversity of opinions (each person must have private information, even if only an eccentric interpretation of the facts)
 - ➋ independence (individuals opinions are not determined by others' opinions)
 - ➌ decentralization (individuals deduce from local knowledge)
 - ➍ aggregation (mechanism to aggregate the responses)

Collective Intelligence

- Surowiecki:

If a group satisfies those conditions, its judgment is likely to be accurate. Why? The answer lies on a mathematical truism. If you ask a large enough group of diverse, independent people to make a prediction, and then average those estimates, the errors each of them makes in coming up with an answer will cancel themselves out. Each person's guess has two components: information and error. Subtract the error and you're left with the information.

Collective Intelligence

- Not so easy case: when an individual decision depends on the collective
- This is the case in the number game and all games of the minority game class
- Let us now play one instance of this game...

Minority/Majority Games

- Pick one of the 3 objects
- Winner(s) is/are whoever picks the most picked object (i.e., is in the majority)



object A



object B



object C

Minority/Majority Games

- If you and m other participants have to select / pick the most beautiful (or any other quality) among n objects, and giving that those selecting the most (or least) selected one win the game, the best strategy is not to select the object that you prefer, but rather the one which you think is the most preferable
- If everybody does this reasoning, the elected object may be neither the one you prefer, nor the one that the majority prefers
- people usually achieve a “third degree of anticipation”, i.e., they anticipate what the majority of people would select and react accordingly

Minority/Majority Games

- Minority / Majority games are common in the real world:
 - 1 free search engine (what is the point in using a search engine nobody uses?)
 - 2 same goes for fax machines
 - 3 route choice
 - 4 technological standards (e.g., betamax vs. VHS or blueray vs. DVD)

In Short

- MAS are not only about collective intelligence (in fact it seldom is)
- It is much more about coordination of decisions made by autonomous, deliberative agents that share an environment

Elements of a MAS

- Environment E (in general, a n -dimensional space where all interactions among agents are represented)
- Set of passive objects O situated in the space, which can be perceived and modified by the agent
- Set A ($A \in O$) of situated agents, i.e., at each given time, each agent is positioned in the space
- Set R of relations between the objects and the agents
- Set O_p of operations with which the agents perceive, produce, consume, transform, or manipulate the objects
- Set O_e of operations over the environment, which represent the effects of O_p in the world, and the reactions over the agents and objects (universal laws)

MAS: Classification

- Number of agents: fix or open
- Granularity: agents described according to their expertise or behavior
- Heterogeneity: one or various classes of agents?
- Organization: from hierarchical to flat societies
- Cooperation or antagonism
- Complexity of interactions: explicitly or implicit communication
- Coupling: intensity of the interactions (can be measured by the relationship between communication acts and inference quantity)

MAS: Associated Concepts

- Organization
- Interaction
- Coordination
- Cooperation
- Communication

Interactions

- Agent – Agent
 - Caused by actions or series of actions that have any effect over the agents' behavior
 - Example: 2 agents wish to occupy the same place (in the space) at the same time
- Agent – Environment
 - Dynamic environment can be a problem for the agent

Interactions

- Types:
 - Via explicit communication
 - Indirect by means of the environment (agent observes a change in the environment)
- Effect: changes in the environment state, as well as in other agents
- Cooperation or antagonism has an important role
- Pre-condition for occurrence:
 - Agents that communicate (explicitly or implicitly) and/or negotiate
 - Agents shall have motivation to interact (e.g., common (sub)goal, use of same resource) or meet physically (e.g., possible collision between robots)

Types of Interaction: Classification

- Regarding goals:
 - Compatible
 - Incompatible:
 - agent A's goal is to reach state p , and B's goal is to reach state q
 - if $p \rightarrow \neg q$ then: $\text{Goal}(A, p) \rightarrow \neg \text{reachable}(\text{Goal}(B, q))$
- Implication:
 - (potentially) cooperative situation when goals are compatible
 - Antagonist when the goals are incompatible

Types of Interaction: Classification

- Regarding task type:
 - Simple: can be individually carried out
 - Complex: can be better carried out if in a joint way
 - Those that can only be carried out jointly (via cooperation)
- Regarding resource:
 - Sufficient
 - Insufficient

Coordination

- Act of manage dependencies among activities
(Malone and Crowston, 1994)

Communication

- Necessary for coordination and cooperation
 - Explicit (speech acts, language, protocols)
 - Implicit (game theory, pheromone, etc.)

Cooperation in MASs

- Individuals have limited abilities and knowledge about the environment and its dynamics
- Resources are not limited
- Improves the performance
- Reduces redundancy
- Conflict handling
- Agents' plans may hinder or damage other agents' plans
- Agents' plans may help or make possible other agents' plans

MASs: Main Messages

- Ideally: try to replicate collective intelligence
- Global behavior emerging from local behavior (and perceptions)
- Representation of a MAS: tuple
- Choice of sensors, type of communication: important questions

Overview

- We will revisit these 3 areas in the light of what we know from MASs:
 - Planning and decision making using MDP
 - Learning
 - Problem solving via constraints satisfaction/optimization
 - (We start by looking at the respective single agent case and then revisit to check what changes in multiagent cases)
- But (before) we now introduce agent-based modeling and simulation (where we can use concepts from MASs)...

ABMS: General Concept

- ABMS: MASs in a simulated environment and in virtual time
- Concept is based on MASs:
 - Entities = Agents
 - Environment composed of:
 - other agents and
 - other environmental entities such as objects, resources
 - Interactions among agents are key
 - ABMS = agents + environment + interactions

Granularity of Representation of the Elements of the System

- Determining the level of detail to be used
- Two basic forms
 - Macrosimulation: system perspective
 - Microsimulation: fine grained entities with distinct state and behaviour
- Also possible: mesoscopic models

Why ABMS ?

- New paradigm that allows more appropriate modelling in:
 - Social science
 - Biology
 - Traffic
 - Other socio-technical systems
 - Emergent phenomena
 - ...
- Provides a more intuitive way of modelling (closer to the real system) thus facilitating the task of non-experts

Relation to Cellular Automata

- CAs: Discrete in space and time
- Spatial locality plays key role
- Uniform rules assigned to cells
- Next state of cell is computed exclusively based on its last state and on the states of neighbors
- Two possibilities
 - Agent = cell
 - Agent = state of cell
- Often: CA as basis for **environment** of agents

Advantages

- Agent-based Models are a powerful paradigm:
 - Dealing with real MASs directly:
real Agent = simulated Agent
 - Allows modelling of adaptation and evolution
 - Modelling of heterogeneous space and population
 - Different levels of observation

Developing an ABMS

- Top down approach:
 - Related to agent-oriented software engineering (AOSE)
 - Takes into account existing/known organizational structures
 - Focus on the phenomena (described on a global level):
start describing the phenomena and later derive agents/behaviors

Developing an ABMS (cont.)

- Bottom up approach:
 - Focus on agents: observing real-world agents in order to derive simulated agents
 - A global behavior may emerge
 - Trial and error
 - Parameter calibration costly
 - Normally requires much more data

Frameworks / Tools

- Netlogo
- Repast
- Swarm
- SeSAm
- MASON
- ...

Overview

- We now revisit these 3 areas in the light of what we know from MASs:
 - **Planning and decision making using MDP**
 - Learning
 - Problem solving via constraints satisfaction/optimization
 - (We start by looking at the respective single agent case and then revisit to check what changes in multiagent cases)

MDP: Introduction

- Generalization of search and planning problems
- Method to decide what to do *today*, given that we will decide again *tomorrow*
- Sequential decision problems where the utility of each agent depends on a sequence of decisions (MDP) and/or on the decision of various agents (MMDP), eventually under partial observation (POMDP)
- Markov assumption: the *effects* of an action taken in a given state depend only on this state (not on the past history)

Related Concept: discounted reward

- *discounted* utility /reward / payoff:
 - what is preferable? 90 dollars now or 100 in one month?
 - 90 now > 100 later ???

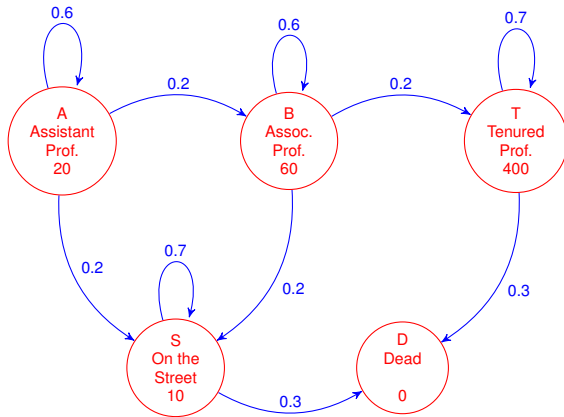
Related Concept: discounted reward (cont.)

- Future payment does not worth the same as payment now (uncertainty regarding being paid, inflation, etc.)
- assuming a discount rate of 0.9, 100 is worth only 90 in the next month!
- Salary of 1000 per month $\neq 1000 + 1000 + 1000 + \dots \rightarrow \infty$
- We cannot say *now* that your salary is infinite!
- Devaluation is given by $(0.9)^N$
- Discount rate γ :
 - total salary: (salary now) + $\gamma \times$ (next month salary) + $\gamma^2 \times$ (salary in 2 months) + $\gamma^3 \times$ (salary in 3 months) + ...

Example 1: Academic Life

- Here: only *state* transitions
- Be:
 - V : value of expected future discounted reward
 - V_A, V_B, V_D, V_S, V_T same in states A, B, D, S, and T
 - $\gamma = 0.9$
- And an automata that considers these states and transitions between them (e.g., assistant professor to associate professor)
- How to compute the future rewards of academic life???

Example 1: Academic Life



MDP

- Formally defined by:
 - set of states $S = \{s_1, s_2, \dots, s_n\}$
 - set of actions $A = \{a_1, a_2, \dots, a_n\}$
 - probability or state transition matrix P where p_{ij} = prob. (next state = j | current state = i)
 - obs.: in general, also considers the action taken
 - rewards for each state $R = \{r_1, r_2, \dots, r_n\}$
 - discount factor $0 < \gamma \leq 1$
- For each step i :
 - state s_i , action a_i , reward r_i
 - random transition to the next state (s_j)
 - all future rewards are discounted by γ

Exact Solution for MDP

- $V^*(s_i)$ is the expected value of the future discounted sum of rewards from state s_i (also denoted as utility or $U(s_i)$)
- $V^*(s_i) = r_i + \gamma[p_{i1} V^*(s_1) + p_{i2} V^*(s_2) + \dots + p_{in} V^*(s_n)]$

- Be:

$$V = \begin{bmatrix} V^*(s_1) \\ \vdots \\ V^*(s_n) \end{bmatrix} \quad R = \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix} \quad P = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ \vdots & \ddots & & \\ p_{n1} & p_{n2} & \dots & p_{nn} \end{bmatrix}$$

- $V = R + \gamma[P \times V]$
- Good news: exact solution !
- Bad news: for 10^2 states, it is necessary to **solve a $10^2 \times 10^2$ system of equations!**

Policy and Optimal Policy for an MDP

- Policy: mapping from states to actions (i.e., what exactly the agent should do in *each* state that is reachable from a given state)
- $\Pi(s_i)$ is the policy for state s_i
- Quality of a policy: given by the expected utility
- Optimal policy $\Pi^*(s_i)$: policy that yields the highest expected utility (there is no better option than follow Π^*)
- For each MDP there exists an optimal policy (how to compute it?)
- Policies can be expressed as tables (but this is not efficient)

Computing the Optimal Policy

- Enumeration and choice: not feasible in real-world problems
- Thus: dynamic programming
- Case 1: finite horizon (fix time k)
 - with a finite horizon, the optimal action can change with time (non-stationarity)
- Case 2: infinite horizon
 - if there is no time limit, then there is no reason to behave in a different manner if the agent is in the same state in different time steps
- Example Powerball Lottery: if you know that the Powerball Lottery will no longer exist from tomorrow on, how do you act? and if it will end in 10 days? in 100? and in case you do not have this information (assuming that the Powerball Lottery will be there forever)?

Computing the Optimal Policy (cont.)

- *Computing policies for infinite horizon is simpler* (due to the stationary nature)
- In infinite horizon, by using *discounted rewards*, the utility of an infinite sequence of actions is *finite*
- When $\gamma < 1$, we have

$$V[s_0, s_1, \dots] = \sum_{t=0}^{\infty} \gamma^t R(s_{i,t}) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{(1-\gamma)}$$

(geometric series)

- $\Pi^* = \arg \max_{\Pi} E \left[\sum_{t=0}^{\infty} \gamma^t R(s_{i,t}) | \Pi \right]$ where E is the expected value

Bellman Equations (1957)

- Be $V^k(s_i)$ the *maximum* sum of the discounted future reward from state s_i at time step k , if agent chooses **action** a
- $V^0(s_i) = r_i$
- In each time step $N + 1$:
$$V^{N+1}(s_i) = \max_a \left[r_i + \gamma \sum_{j=1}^n p_{ij}^a V^N(s_j) \right]$$
- **one equation for each state!**
- The expected utilities of the sequence of states ($V[s_0, s_1, \dots]$) are the unique solution of the set of Bellman equations

Steps for Solving Bellman Equations

- Solve $|S|$ simultaneous equations
- Equations are **non linear**
- Hard to solve analytically; thus iterative solution:
 - 1 initialize V^0 with arbitrary values
 - 2 for each S , compute $V(S)$ from $R(S)$ and $V^0(S)$
 - 3 use these new utility values to update V
 - 4 repeat steps 2 and 3 until V converges
- This equilibrium is the unique solution

Solving MDP via Value Iteration

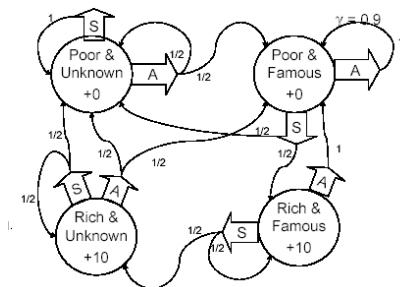
- compute $V^1(s_i)$ for each i
- compute $V^2(s_i)$ for each i
- \vdots
- $V^k(s_i)$ for each i
- \vdots
- ... until convergence!
- when?
 - when $\max_i |V^{N+1}(s_i) - V^N(s_i)| \leq \epsilon$

Value Iteration: Algorithm

```
function value_iteration(S,P,R, $\gamma$ , $\epsilon$ )  
  repeat  
     $V[ ] \leftarrow R[ ]$ ;  $\delta \leftarrow 0$   
    for each state  $s \in S$  do  
      do  $V'[s] \leftarrow R[s] + \gamma \max_a \sum_{s'} p_{ij,a} V[s']$   
      if  $|V'[s] - V[s]| > \delta$  then  
         $\delta \leftarrow |V'[s] - V[s]|$   
      end if  
    end for  
  until  $\delta < \epsilon \times \frac{1-\gamma}{\gamma}$   
  output  $V$ 
```

Example with Actions

- you are the owner of a company
- actions at each time step: invest (S) or advertise your product (A)
- be the states and transitions graph



Example with Actions (cont.)

- applying the algorithm (with $\gamma = 0.9$):

k	$V^k(PU)$	$V^k(PF)$	$V^k(RU)$	$V^k(RF)$
0	0	0	+10	+10
1	0	4.5 ⁽¹⁾	14.5	19
2	2.03	8.55 ⁽²⁾	16.53	25.08
3	4.76	12.2 ⁽³⁾
⋮				

$$\textcircled{1} \quad V^1(PF) = R(PF) + \gamma \times \max \left\{ \underbrace{\frac{1}{2} V^0(PU) + \frac{1}{2} V^0(RF)}_S, \underbrace{1 V^0(PF)}_A \right\} = 0 + 0.9 \max \left\{ 0 + \frac{10}{2}, 0 \right\} = 4.5$$

$$\textcircled{2} \quad V^2(PF) = R(PF) + \gamma \times \max \left\{ \frac{1}{2} V^1(PU) + \frac{1}{2} V^1(RF), 1 V^1(PF) \right\} = 0 + 0.9 \max \left\{ 0 + \frac{19}{2}, 4.5 \right\} = 0 + 0.9 * 9.5 = 8.55$$

$$\textcircled{3} \quad V^3(PF) = R(PF) + \gamma \times \max \left\{ \frac{2.03}{2} + \frac{25.08}{2}, 8.55 \right\} = 12.2$$

Sequential Decision Problems

- MDP's involve two main questions
 - 1 is horizon finite or infinite?
 - 2 how to find the optimal behavior?
 - value iteration
 - policy iteration

Horizon

- Question 1, finite horizon:
 - there is a fixed time n after which the interaction with the environment ends
 - formally: $U([s_0, s_1, \dots, s_{n+k}]) = U([s_0, s_1, \dots, s_n]), \forall k > 0$

Horizon (cont.)

- Question 1, infinite horizon:
 - with no fixed time limit
 - there is no reason to behave differently in a given state at different times
 - optimal behavior depends only on the current state
 - optimal policy is stationary
 - simpler case

Optimal Behavior

- Question 2: How to find the optimal behavior?
 - an option: list all behaviors and choose the one(s) with the highest possible value for each initial state
 - not feasible!
 - better option: calculate value functions
 - compute the so-called optimal value function and then the optimal behavior
 - the optimal value of the state s , $V^*(s)$, is the one that obtains the highest expected value when the agent starts in state s
 - this means: compute the utility itself (U or V) and, with this, the optimal behavior

POMDP

- Relaxation of the hypothesis “ environment is completely observable ”
 - agent does not necessarily know what state it is in
 - how to execute the recommended policy for the state it is in?
 - state utility s and optimal action in s depend not only on s but also on *how much the agent knows* when it is in s
- POMDP is more difficult than MDP
- POMDP is more realistic than MDP

Multiagent MDP

- An agent trying to decide sequentially while **other agents** are in the environment trying to do the same
- MDP for more than one agent is a multiagent MDP (MMDP), also known as (**stochastic game** or SG)
- Base: game theory!

Stochastic Games

- A stochastic game (SG) is defined by a set of states \mathbf{S} with a stage game defined for each state $s \in \mathbf{S}$
- In each state s , player i can choose actions from set $\mathbf{A}_i(s)$
- One of the stage games is played at each time $t = 0, 1, 2, \dots$
- The actions played by the players determine not only their immediate rewards (payoffs), but also the probability of transitioning to any given state at the next time step
- This means: given that 2 players **1** and **2** are in state s and choose actions $a_1 \in \mathbf{A}_1$ and $a_2 \in \mathbf{A}_2$, these players receive rewards $r_1(s, a_1, a_2)$ and $r_2(s, a_1, a_2)$; also, the probability they find themselves in state s' in the next time step is $p(s'|s, a_1, a_2)$

Stochastic Games (cont.)

- Definition: a strategy is called a Markov strategy if the behavior of a player at time t depends only on the state s ; a pure Markov strategy specifies an action $a(s)$ for each state $s \in \mathbf{S}$

Stochastic Games (cont.)

- Henceforth the following assumptions are made:
 - ① the length of the game is not known to the players (i.e., the horizon is infinite)
 - ② rewards and transitions are time-independent
 - ③ strategies are Markov

Multiagent MDP

- Generalization of an MDP for n agents is given by the tuple $\text{MMDP} = (\mathcal{I}, \mathbf{S}, \mathbf{A}, \mathcal{P}, \mathcal{R})$
 - set of agents $\mathcal{I} = 1, \dots, i, \dots, n$
 - set of states \mathbf{S}
 - set of actions \mathbf{A}
 - reward function \mathcal{R}
 - transition probability matrix \mathcal{P}
- Also (depending on the modeling):
 - $\mathbf{A} = \times_{i \in \mathcal{I}} \mathbf{A}^i$ and $\mathbf{S} = \times_{i \in \mathcal{I}} \mathbf{S}^i$
- Depending on the actions performed by the agents, a transition occurs that determines a new state, i.e. a new reward matrix
- We already know the rest (from the theory of repeated games)

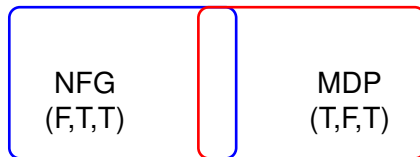
MDP and GT

(1 agt ?, 1 tmstep ?, fully obsv. ?)

NFG
(F,T,T)

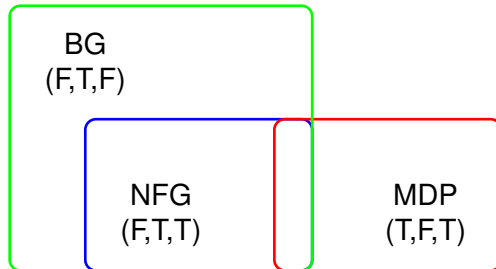
MDP and GT

(1 agt ?, 1 tmstep ?, fully obsv. ?)



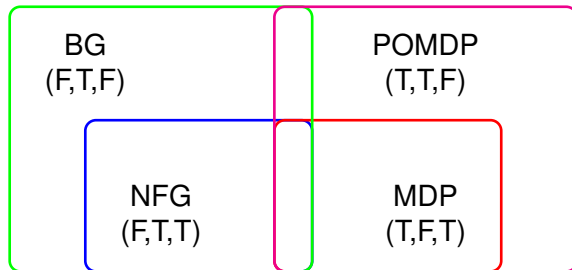
MDP and GT

(1 agt ?, 1 tmstep ?, fully obsv. ?)



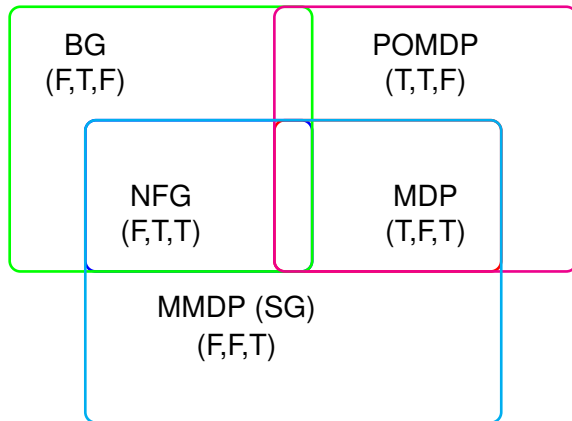
MDP and GT

(1 agt ?, 1 tmstep ?, fully obsv. ?)



MDP and GT

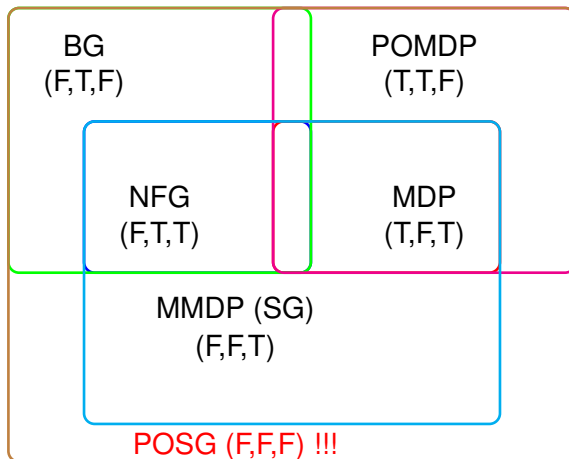
(1 agt ?, 1 tmstep ?, fully obsv. ?)



MDP and GT

(1 agt ?, 1 tmstep ?,

fully obsv. ?)



MDP & POMDP : Suggested Reading

- Givan, Bob. MDP Tutorial – definitions and extensions, Dagstuhl, November, 2001
- Jensen, PA and Bard, JF. Operations Research: Models and Methods. John Wiley & Sons, NJ, 2003
- Muñoz-Avila, H. Uncertainty and Utility (for Lehigh University CSE 395/495 Spring 2003)
- Russell, SJ and Norvig, P. Artificial Intelligence: a Modern Approach. Prentice Hall, NJ, 2003 (2nd ed.)
- Leslie Pack Kaelbling, Michael L. Littman and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence, Vol. 101, 1998.
- Goldman, C.V. and Zilberstein, S. (2004). Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis, Volume 22, pages 143-174.

Overview

- We now revisit these 3 areas in the light of what we know from MASs:
 - Planning and decision making using MDP
 - **Learning**
 - Problem solving via constraints satisfaction/optimization
 - (We start by looking at the respective single agent case and then revisit to check what changes in multiagent cases)

Machine Learning

- Numerous methodologies
 - symbolic, connectionist, etc.
 - tree induction
 - genetic algorithms and evolutionary computation
 - neural networks
- None is directly linked to **decision**
- (and also not to autonomy)
 - How to act?
 - What action to take in each situation?

Machine Learning

- We have already seen how planning in nondeterministic environments (i.e., with uncertainty) can be realized by calculating and using optimal policies
 - this assumes that the agent has (knows) the **correct** model of the environment (i.e. knows P and R)
- Now we will see that an agent who does not know such a model can learn about this
 - how? interacting with the environment!

Motivation for RL

- Given a goal to be achieved by the agent:
 - we do not want to have to detail, step by step, how the agent should act
 - we want the agent **to find out** how to act in each situation
 - **Autonomy**
- What information do we give to the agent?
 - We do **not** tell the agent if its decision is right or wrong
 - Rather: we give only an indication about the **instant quality for each decision** → **a reward sign**

Motivation for RL (cont.)

- **Learning by reinforcement**
 - Can not be considered supervised or unsupervised learning
 - Can be imagined as **learning with a critic**
- Initially inspired by the experiments of psychology
- Classic conditioning (Pavlov's dogs)
- Law of effect (Thorndike / Skinner)
 - Stimulus-reinforcement associations (positive or negative)
 - **Positive** reinforcements strengthen the probability of a behavior
 - **Negatives** reinforcements weaken such probability

Motivation

- Algorithms for **reinforcement learning**
 - Given a **world description**, agent learns how to act optimally
 - list of possible **actions**
 - quality indication of each action in each situation → sign of **reward** for each action in each state
 - We allow the agent **to practice** and **learn** (trial-and-error)!

Formalization as MDP

- How do we formalize this? **Markov models**
 - generalization of search and planning problems
 - methods for deciding what to do **now**, given that **after** we will decide again
 - sequential decision problems (MDP, MMDP, POMDP ...)
 - Markov hypothesis: **effects** of an action taken in a given state depends only on this state and not on past history

Formalization as MDP

- Markov models

- set of **states** \mathcal{S} ;
- set of **actions** \mathcal{A} ;
- set of “preferences”, or **reinforcement signs**
 - tell the agent how good (or bad) each action is, in each situation where it can be applied
 - $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$;
- **probabilistic transition function** $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$
 - $T(s, a, s') =$ transition probability $s \rightarrow s'$ given action a

Formalization as MDP

- Example 1:

- **Environment:** You are in **state A** and have **4** possible actions
- **Agent:** I choose action **2**
- **Environment:** You received reinforcement sign of **+7** points. You are in **state B** and have **5** possible actions
- **Agent:** I choose action **5**
- **Environment:** You received reinforcement sign of **-2** points. You are in **state F** and have **2** possible actions
- **Agent:** I choose action **2**
- **Environment:** you are in **state A** and have **4** possible actions
- **Agent:** I choose action **2**
- **Environment:** You received reinforcement sign of **+7**
- ...

Formalization as MDP

● Example 2:

- **Environment:** You are in **state A** and have **4** possible actions
- **Agent:** I choose action **2**
- **Environment:** You received reinforcement sign of **+7** points. You are in **state B** and have **5** possible actions
- **Agent:** I choose action **5**
- **Environment:** You received reinforcement sign of **-2** points. You are in **state F** and have **2** possible actions
- **Agent:** I choose action **2**
- **Environment:** you are in **state A** and have **4** possible actions
- **Agent:** I choose action **2**
- **Environment:** You received reinforcement sign of **+1** points. You are in **state C** and have **3** possible actions
- ...

Formalization as MDP

- Goal: find an optimal policy π^*
 - for every possible state s : indicates which is the **best action**
 - this means: $\pi(s) = a$
 - what does “best action” mean?
 - leads the agent to states from which it can earn more “points” (signs) of reinforcement
 - .. (assuming the agent continues to act as best it can)
 - reinforcement points the agent will get in the future are the *expected* ones
 - transitions are probabilistic
 - worse: reinforcement may not be deterministic !!! (What is the difference between previous examples 1 and 2?)

Formalization as MDP

- How to measure **how good is each situation** in which the agent can be?
- We associate each state s with a value $V(s)$
 - V = reinforcement that we hope the agent will receive in the future
 - given that the agent starts in s and continues selecting optimal actions
- We have already seen how to compute optimal policies (value iteration, policy iteration)

Problems

- Problem with policy iteration and value iteration:
- **Agent needs to know several things**
 - $T(s, a)$ (or P), for all s and all a
 - $R(s, a)$, for all s and all a
 - in other words, agent needs to know the **environment model**
 - the agent should not need to know everything about his environment, before it can act (**babies!**)
- Can we estimate $V(s)$ without having the model?

RL Types

- According to the type of agent design, a type of learning

agent design	know	learn	use
utility-based	T	$R \rightarrow U$	U
Q-learning-based	-	$Q(s, a)$	Q
reflex agent	-	π	π

- Besides this, passive or active learning:
 - passive: agent follows the prescribed policy and only explores if, due to the stochasticity in the environment, it visits a certain state "by accident"
 - active: agent greedy uses the prescribed policy (as in the passive) but exploits other alternatives with a low probability

Q-Learning

- Almost the same thing as calculating V , but ...
 - instead of calculating a V value for each **state**
 - i.e. "how good it is to be in this state"
 - calculates a value $Q(s, a)$, for each s and each a
 - i.e. "how good is to perform this action in this state"
 - the agent perceives its state s , chooses an action a and receives a reward r
 - .. then updates its estimate of Q value for the pair $\langle s, a \rangle$
 - $$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') \right)$$

Q-Learning

- And how does the agent choose an action?
- If it is in s and there it can do 4 actions:
 - has estimates for how good it is to perform each of these actions
 - $Q(s, a_1)$, $Q(s, a_2)$, $Q(s, a_3)$ and $Q(s, a_4)$
- It chooses the largest!
- $\pi(s) = \operatorname{argmax}_a Q(s, a)$
- Direct relationship of Q with V : $V^*(s) = \max_a Q^*(s, a)$

Q-Learning

- Shall the agent always choose the best action?
 - No! because it would then not explore further!
- There are several ways to balance exploration and exploitation (of the best perceived action): known as **exploration-exploitation dilemma**
 - epsilon-greedy: select the best action (greedy action selection) most of the time but, with a small probability (ϵ) select a random action
 - other methods such as softmax / Boltzmann exploration

Problems

- Reinforcement learning has several problems
 - **non-stationary environments**
 - agent “forgets” already computed policies (only the last one is saved)
 - solution: multiple models, one for each “state” of the environment
 - **non fully observable environments**
 - the agent is not sure which is the current state
 - solution: POMDP, etc. (based on beliefs)
 - **exponential number of states or combination of states–actions**
 - solution 1: state abstraction
 - solution 2: hierarchical reinforcement learning

Multiagent reinforcement learning

- Agent tries to solve a problem / tries to learn while **other agents** are in the environment trying to solve their learning problems (related or not)

Multiagent reinforcement learning

- Problems with MARL (multiagent RL)
 - Fundamental hypotheses of RL are violated
 - Interaction in dynamic environment (dynamics come from other agents reacting to the same environment)

Multiagent reinforcement learning

- Formalization as MMDP / SG:
 - There is no known general solution (e.g. general sum games)
 - Combinatorial explosion on states and actions
 - Set of states $S_1 \times S_2 \times \cdots \times S_n$
 - Set of actions $A_1 \times A_2 \times \cdots \times A_n$
 - $S_1 \times S_2 \times \cdots \times S_n \times A_1 \times A_2 \times \cdots \times A_n \rightarrow \mathbb{R}$
 - It is necessary to visit an exponential number of combinations of pairs S, A

Multiagent reinforcement learning

- Some ways to deal with the aforementioned problems:
 - Ignores the presence of other agents learning and applies simple RL
 - Perceives the presence of other agents but seeks an independent convergence
 - Adapts to the other agents seeking the optimality
 - Agents can be heterogeneous ...

Multiagent reinforcement learning

- A first idea is for each agent to learn ignoring the others
- Agent uses Q-Learning as if it were unique in the environment
 - Treats other agents as part of the environment
 - It works if the behavior of the other agents is fixed
 - If other agents also learn, agent policy i has to adapt to the adaptation of others
 - This is why multiagent RL is so hard

Multiagent reinforcement learning

- Types:
 - Cooperatives / team game: $\rho_1 = \rho_2 = \dots = \rho_n$
 - Competitive / zero-sum game: $\rho_1 = -\rho_2$
 - Competitive / non-zero-sum game: more general case

Multiagent reinforcement learning

- Purely cooperative:
 - Global state: includes states of all agents
 - Set of actions that considers all agents
 - Value $Q(s, a)$ would be $Q(< s_1, s_2, \dots, s_n >, < a_1, a_2, \dots, a_n >)$
 - **Problem:** Q Table **too big** and the communication should be **unrestricted!**

Multiagent reinforcement learning

- Example of Q table explosion:
 - n agents, m actions each and k states $\rightarrow m^n * k$ entries in Q table
 - simple problem: consider 3 agents in a grid of size 10x10 cells, with 4 possible actions (move to: up, down, right, left)
 $\rightarrow 4^3 * 100^3 = 64 \text{ million}$ entries in Q table!

Multiagent reinforcement learning

- Alternatives to full description:
 - share information (perceptions, policies or episodes)
 - sparse Q table
 - unification of similar states (using function approximation)

Sparse and Cooperative Q-learning

- Hypothesis: in several problems agents need to coordinate their actions in only a few states, while in others they can act independently
- Idea: groups of agents learn how to solve a task in a cooperative way **when the coordination requirements are known beforehand**
- Details: Oliveira and Bazzan (2009); Bianchi and Bazzan (2012)

Multiagent reinforcement learning

- Seen by the Game Theory ...
 - a joint strategy is **Pareto-optimal** if there is no strategy that increases *payoffs* of all agents / players
 - a strategy x_i of an agent is **dominant** if it is always the best, regardless of the action of the other agents
 - x_i is the best answer to x_{-i} if it maximizes the *payoff*

Multiagent reinforcement learning

- Seen by Game Theory ... (continued)
 - a joint strategy x is an **equilibrium** if the strategy of each agent is simultaneously the best response to the strategy of the others
 - that is, there is no incentive to deviate: Nash equilibrium
 - **at least one** Nash equilibrium always exists but **several** may exist and / or be difficult to calculate ...

Multiagent reinforcement learning

- Single-agent systems (with MDPs):
there is always a deterministic optimal policy
- Multi-agent systems (with MDPs):
deterministic policies can be dominated (not interesting)
- How to define an optimality criterion?
 - Agent performance according to strategies of other adaptive agents
 - *Nash equilibrium*

Recurring discussion on MARL

- Criticism of Shoham et al (Stanford):
If multiagent learning is the answer, what is the Question?
- Answer from Peter Stone (University of Texas)
Multiagent learning is not the answer. It is the Question.
- See AI Journal 171(7), 2007

Remarks on MARL

- Multiagent learning models / algorithms consider small scenarios with 2 agents
- Almost always when there are more than 2 agents, these are completely homogeneous
- How to reduce state space?
- Open problems: co-learning (e.g. heterogeneous agents), nondeterministic environments, large number of agents and / or states
- Which techniques could be used together?

Remarks on MARL (cont.)

- Multiagent scenarios much harder than single agent ones
- Adaptation is key
- Nature inspiration (e.g. self organization for task allocation in the rescue scenario)
- Challenges

Suggested Reading

- Kaelbling et al., 1996: Reinforcement Learning: A Survey
- Artificial Intelligence Journal (Volume 171, Issue 7, Foundations of Multi-Agent Learning)
 - Yoav Shoham, Rob Powers and Trond Grenager. *If multi-agent learning is the answer, what is the question?*
 - Peter Stone. *Multiagent learning is not the answer.* It is the question.
- Sutton and Barto 1998: *Reinforcement Learning: An Introduction.*
Book available at:

<http://www.cs.ualberta.ca/~sutton/book/ebook/the-book.html>

Overview

- We now revisit these 3 areas in the light of what we know from MASs:
 - Planning and decision making using MDP
 - Learning
 - **Problem solving via constraints satisfaction/optimization**
 - (We start by looking at the respective single agent case and then revisit to check what changes in multiagent cases)

CSP and Beyond

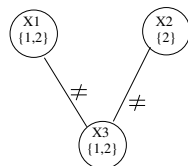
- We now discuss problem solving using constraint satisfaction...
- ... and extension to the multiagent case
- In the beginning... it was... CSP (constraint satisfaction problem)
 - CSP \rightarrow COP (constraint optimization problem)
 - CSP \rightarrow DisCSP (distributed CSP)
 - DisCSP \rightarrow DCOP (distributed COP)

Problem Solving

- CSP: based on search algorithms that take advantage of some sort of structure about states of the problem in order to solve it
- A CSP is defined by a set of variables (V_i) (and their domains) (D_i), and by a set of constraints
- A solution is an assignment of values to each variable, so that no constraint is violated

Representation

- It is helpful to represent/visualize a CSP as a constraint graph
- Example: find a value for each variable ($\{x_1, x_2, x_3\}$) within their domains ($\{1, 2\}, \{2\}, \{1, 2\}$), observing the constraints ($\{\{x_1 \neq x_3\}, \{x_2 \neq x_3\}\}$)
- Solution (unique, in this case): assignment
 $\mathcal{A} = \{\langle x_1 = 2 \rangle, \langle x_2 = 2 \rangle, \langle x_3 = 1 \rangle\}$



How about if there is no unique solution?

- some problems have no solution at all (it is not possible to satisfy all constraints) !
 - What can be done here: *minimize* the number of broken/relaxed constraints, or minimize some cost function
- other set of problems have more than one solution
 - what can be done here: choose the *best* (in terms of some cost function) among the solutions
- Both these issues contributed to the COP (constraint optimization problems) area where we seek to *optimize* some cost function

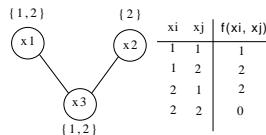
COP

- Find the *best* solution
- Increases the complexity (optimization rather than satisfaction)
- Formalization of a COP:
 - besides variables, their domains, and constraints
 - for COP: $\mathcal{F}(\mathcal{A})$
 - objective function that maps each solution (set of assignments) to a numerical values
 - normally an aggregation of cost functions f
 - solution: a complete assignment \mathcal{A}^* corresponding to a optimal value for the objective function

COP

● Example

- find a value for each variable under the constraints so that the objective function is minimized
- $\mathcal{F}(\mathcal{A})$ is given by the sum of various f functions
- $\mathcal{F}(\mathcal{A}) = f(x_1, x_3) + f(x_2, x_3)$
- solution: a complete assignment
 $\mathcal{A}^* = \{\langle x_1 = 2 \rangle, \langle x_2 = 2 \rangle, \langle x_3 = 2 \rangle\}$

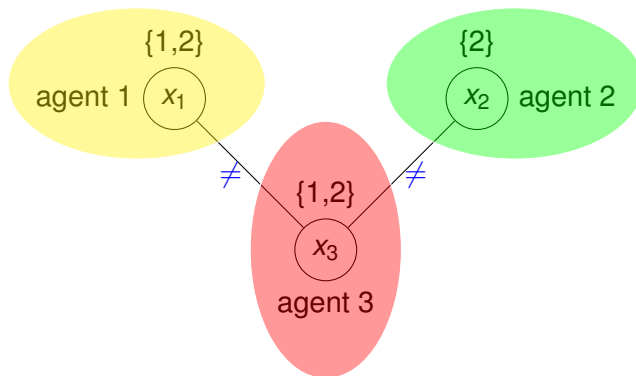


Multiagent AI – DisCSP/DCOP

- Distributed Constraint Optimization Problem
 - similar to CSP / COP
 - Instead of centralized: distributed
- Why distributed?
- If centralized:
 - Single point of failure
 - Bottleneck
 - Not possible to centralize all information
 - Network limitation or privacy
- Need to formalize and solve in a distributed way

Multiagent AI – DisCSP/DCOP

- DisCSP and DCOP
 - Variables and constraints assigned to agents
 - Increased communication



DisCSP: Formalization

- m agents $1, 2, \dots, m$
- Each variable x_j belongs to agent i : $\text{belongs}(x_j, i)$
- Constraints are thus distributed among the agents
- Agent l knows constraint p_k : $\text{known}(p_k, l)$
- DisCSP solution:
 - $\forall_i \forall_{x_j}$ where $\text{belongs}(x_j, i)$, the value of x_j is d_j
 - $\forall_l \forall_{p_k}$ where $\text{known}(p_k, l)$ is true, p_k is true when $x_j = d_j$

Distributed Constraint Optimization – DCOP

- Formalization is similar to DisCSP
- But: besides knowing the constraints, agents must also know the cost functions related to these constraints

DisCSP / DCOP: algorithms

- Agents communicate (via exchanged messages) in order to coordinate and find a solution
- Algorithms
 - DisCSP
 - Asynchronous Backtracking (AB)
 - DCOP
 - Asynchronous Distributed Optimization (Adopt)
 - Distributed PseudooTree Optimization (DPOP)

BT Algorithm (synchronous)

- ➊ Order all agents (some lexicographic ordering)
- ➋ The agent with highest priority (e.g., A_1) assigns a value for its variable
- ➌ A_1 uses a message to send its assignment to the agent with the highest priority after itself (e.g., A_2)
- ➍ All agents wait for a message – containing a partial solution – coming from the agent with highest priority below itself
- ➎ When A_i receives a message, it tries to assign a value for its variable, so that no constraint is violated (i.e., all constraints are satisfied)
- ➏ If a such assignment is found, then A_i adds it to the message and sends it to agent A_{i+1}
- ➐ If not, A_i returns the same message (as received) back to A_{i-1}
- ➑ A solution is reached when the last agent (A_n) finds an assignment for its variable
- ➒ A non-solution state is declared when the agent with highest priority receives a message back and has no more values to assign to its variable (i.e., the whole domain was already assigned) but no solution was found

ABT Algorithm (asynchronous)

- Two types of messages:
 - 1 **ok?** message: this message asks an agent if a given partial solution is acceptable
 - 2 **nogood** message: this message informs an agent that has sent a partial solution that a constraint violation was found (i.e., the tried value is not good)
- **ok?** message: higher to lower priority
- **nogood** message: lower to higher priority

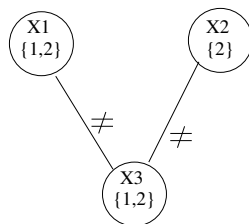
ABT Algorithm (cont.)

- Uses a constraint graph
- Each agent, concurrently, assigns a value to its variable and uses an **ok?** message to send this value to the agents that are connected to it
- Then the agent waits for an answer
- The agent that has received a **nogood** message (evaluator agent) keeps a so-called **agent_view** (set of values assigned by the agents with which this agent has connection) and have higher priority
- The evaluator agent adds the value received in the **ok?** message to its **agent_view**
- Next, it evaluates whether the assignment for its own variable is consistent with its **agent_view**

ABT Algorithm (cont.)

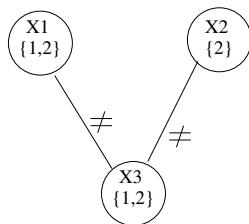
- An assignment is consistent with an **agent_view** iff all constraints that are known by the agent remain satisfied when the agent assigns a value to its variable
- If an assignment is not consistent with the **agent_view**, then the agent tries to change the value of the variable
- If the agent is unable to find a value that is consistent, then it returns a **nogood** message

Example Yokoo



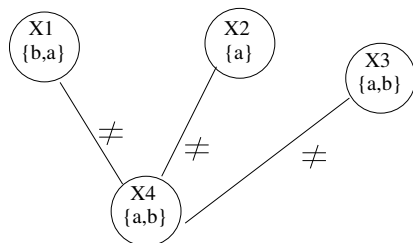
- **X1** and **X2** send messages with values: (**ok?** (X1,1)) and (**ok?** (X2,2))
- **X3** creates its **agent_view**: (X1,1),(X2,2)
- **X3** finds out that this is a nogood
- **X3** sends a **nogood** to **X2**
- a new link between **X2** and **X1** is then created (only **X2** inserts this new link to its constraint graph)

Example Yokoo (cont.)



- $\{(X1,1)\}$ is a **nogood** because there are no more options of values for **X2**
- thus **X2** sends a **nogood** message

Example Bessière



- What if there is no solution?

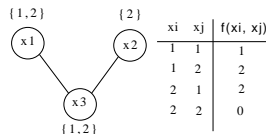
Remember: COP

- Finds the *best* solution
- Increases the complexity (optimization rather than satisfaction)
- Formalization of a COP:
 - besides variables, their domains, and constraints
 - for COP: $\mathcal{F}(\mathcal{A})$
 - objective function that maps each solution (set of assignments) to a numerical values
 - normally an aggregation of cost functions f
 - solution: a complete assignment \mathcal{A}^* corresponding to a optimal value for the objective function

Remember: COP

● Example

- find a value for each variable under the constraints so that the objective function is minimized
- $\mathcal{F}(\mathcal{A})$ is given by the sum of various f functions
- $\mathcal{F}(\mathcal{A}) = f(x_1, x_3) + f(x_2, x_3)$
- solution: a complete assignment
 $\mathcal{A}^* = \{\langle x_1 = 2 \rangle, \langle x_2 = 2 \rangle, \langle x_3 = 2 \rangle\}$



DCOP

- How to do this in a distributed way using agents?
- Popular algorithms: Adopt and DPOP (and their variants)
- Basis:
 - Establish priorities among the agents by means of a search tree (depth search)
 - Nodes in different branches are different sub-problems that can be solved independently (asynchronously)

DisCSP and DCOP: applications

- Various real world problems which are either inherently distributed or where privacy is an issue:
 - sensor network: choose (assign) a value (e.g. on/off) in a distributed way
 - meeting schedule: choose (assign) a value (e.g. meeting time slot) in a distributed way so that no participant (agent) needs to disclose all information

DisCSP and DCOP

- Overview methods : Issue 161 – Artificial Intelligence Journal (AIJ)

The “old days” motivations for multiagent learning

- In MAS: learning helps agents make decisions
- However: now-a-days in the multiagent systems (MASs) community, learning is immediately “translated” into reinforcement learning (RL)
- This is a change in the “old days” motivations for multiagent learning, i.e., interactions between MASs and machine learning (ML) should be manifold

MASs and ML were closer

- Interactions between MASs and ML should be manifold
- Idea has appeared as early as in the 1990s
 - Brazdil *et al.* (1991); Brazdil and Muggleton (1991); Davies and Edwards (1995); Sian (1991)
 - European Working Session on Learning (EWSL-91)
- However: we still keep saying that **multiagent learning (MAL)** should be more than **multiagent reinforcement learning (MARL)**:
 - Cao *et al.* (2007); Stone (2007); Tuyls and Weiss (2012)
- Haven't the 1990s ideas materialized ?
- An still open question is: what exactly is meant with MAL?

“Old days” Agenda for MAL

- Synergy between MASs and ML was of a broader nature, and the motivations for and questions around MAL were different
- Brazdil and colleagues in 1991: “Two rather different questions can be formulated in the context of studying ML in multi-agent systems. First, how can multi-agent systems benefit from machine learning. Second, how can machine learning benefit from considering multi-agent set-up. As multi-agent systems are by nature complex, machine learning techniques may be the only way to achieve a robust and versatile system.”

Compilations and reflections on MAL

- Brazdil and colleagues: see before
- Stone and Veloso (2000): survey on existing MAS techniques, emphasizing the **use of ML**
 - **ML** techniques: can be directly **applied in multiagent scenarios**, either by delimiting a part of the domain that only involves a **single agent**, or by focusing on **learning issues that arise because of the multiagent aspect of a given domain**
 - They listed learning approaches (before 2000) for some classes of problems along two dimensions:
homogeneity and **communication**

Stone and Veloso Survey

- **homogeneous non-communicating** agents:
 - **NRL**: stigmergy; local knowledge; game theory
 - **RL**: environment independent reinforcement acceleration (EIRA); Q-learning for foraging
- **heterogeneous non-communicating** agents:
 - **NRL**: genetic programming; genetic algorithms; case-base reasoning; game theory; competitive co-evolution
 - **RL**: multiagent RL for adaptive load balancing; Q-learning; Minimax-Q

Stone and Veloso Survey

- **homogeneous communicating** agents:
 - **NRL**: distributed sensing
- **heterogeneous communicating** agents:
 - **NRL**: learning social behaviors; cooperative co-evolution; Bayesian learning
 - **RL**: multiagent Q-learning

Distributed ML: Motivations Related to MASs

- As it seems, the “old days” agenda on putting together ML and MASs forming multiagent learning (MAL), which was much broader than just using RL, was kind of abandoned by the MASs community
- In the meantime, **what has happened in the ML community** ?
- Part of that agenda now runs under distributed ML
- This deals with a large amount of data, data that is distributed over many locations, and privacy issues
- **Distribution and privacy issues are opportunities for the interaction between agents and ML !**

Distributed ML

- Foundations: ensemble learning
- Idea: build a set of classifiers, e.g., by training each one on different subsets of data and later combining them
- Ensemble learning suits well a distributed, MASs environment, especially regarding horizontal (regarding instances) partition


Distributed ML

- Characteristics of ensembles:
 - ① use of different learning processes to train classifiers with different learning biases (**diversity of methods**);
 - ② combination of different answers for the same problem (**diversity in solutions**);
 - ③ ability to deal with large and distributed data sets (**partitions of data can be allocated to different processors**);
 - ④ profit from **distributed expertise (diversity in data)**
- Algorithms for distributed ML are based on some kind of integration or combination mechanism that operates on the partial results found by distributed actors (e.g., classifiers)
- **Arbiters and/or broadcasting** of the local classifier are frequently employed
- **Many of these issues are tackled by MASs**

Integration of MASs and distributed ML

- For integration MASs and distribute ML: two particular points are important
 - ① each ML algorithm has a bias: implies a **preference for certain generalizations over others**
 - ② different ML techniques use **different representations** of the attributes; it is hardly possible to have a common representation (at least not without losing information)
- What are the implications for MASs ?

Integration of MASs and distributed ML

- For classification tasks: **agent-based solutions to reach the “ensemble effect”**
- Agents can have **different expertise** or have access to **different data**, they may **work together towards a consensus** and more reliable prediction
- To address privacy and representation issues: **agents do not exchange raw data** (only the outputs) using an abstract representation (hypothesis over classes, etc.)
- Distributed ML is performed by first generating local models based on the distributed data analysis, and then adopting a strategy to **combine them into a global model**
- Collaborative learning: agents share information and perform **negotiation to devise a coherent global model**
- Second step – **combination of solutions** – is where MASs.  techniques have more room to be applied

Integration of MASs and distributed ML

- Applications of agent-based ML are those dealing with:
 - heterogeneous and distributed data
 - data where ethical and privacy issues arise: medical data (diagnosis, treatment details, laboratory data, among others, which are provided by multiple and independent sources), fraud detection in financial organizations, etc.

Agent-based ML

● Issues

- heterogeneous and/or self-interested agents: Kargupta *et al.* (1999), Tozicka *et al.* (2007)
- exchange information during learning: Modi and Shen (2001); Santana *et al.* (2006); Bazzan (2013); Chaimontree *et al.* (2012); Wardeh *et al.* (2012)
- privacy: Stolfo *et al.* (1997); Silva *et al.* (2005); Modi and Shen (2001); Recamonde-Mendoza and Bazzan (2016)
- feature and parameters selection: Zhang and Wang (2012); Babu *et al.* (2009)
- negotiation: Chaimontree *et al.* (2012); Emele *et al.* (2012)

Agent-based ML

● Techniques

- diversity of algorithms (classification): Kargupta *et al.* (1999); Recamonde-Mendoza and Bazzan (2016)
- diversity of algorithms (clustering): Chaimontree *et al.* (2010)
- hierarchical clustering: Kargupta *et al.* (1997)
- cluster ensemble: Agogino and Tumer (2006)

● Applications

- intrusion detection: Stolfo *et al.* (1997)
- bioinformatics: Bazzan (2009); Recamonde-Mendoza and Bazzan (2016)
- tracking / trajectories: Zhang and Wang (2012)

ML in and for MASs

● Issues

- heterogeneous agents: Nunes and Oliveira (2004); Brahmi *et al.* (2012); Bekkerman *et al.* (2007)
- negotiation: Hindriks and Tykhonov (2008)
- opponent modeling: Hindriks and Tykhonov (2008)

● Techniques

- multi-objective clustering: dos Santos *et al.* (2009)
- classification / regression: Emele *et al.* (2012); Şensoy *et al.* (2013)
- clustering: Pan *et al.* (2013); Brahmi *et al.* (2012); Ogston *et al.* (2003); Garruzzo and Rosaci (2008); Piraveenan *et al.* (2008)
- biologically inspired: Alam *et al.* (2009); Lemmens and Tuyls (2009); dos Santos and Bazzan (2012)

ML in and for MASs

● Applications

- intrusion detection / web: Alam *et al.* (2009); Bekkerman *et al.* (2007); Brahmi *et al.* (2012)
- traffic / trajectories / maze: Fiosins *et al.* (2012); Lemmens and Tuyls (2009); Nunes and Oliveira (2004); Urban *et al.* (2010)
- sensor networks: Piraveenan *et al.* (2008)
- supply chain management / TAC: Kiekintveld *et al.* (2007)

Main characteristics

- Majority of works: agents encapsulating ML algorithms in order to enhance data mining in applications such as credit card fraud detection system, intrusion detection, market segmentation, sensor networks, customer profiling, bioinformatics, etc.
- Many use classification or clustering algorithms
- Let us look at some of them...

Agent-Based Classification

- Modi and Shen (2001): decentralized algorithms based on **collaborative learning for distributed classification tasks**
 - agents learn their models individually based on a set of local features, collaborating with each other by exchange of information during the learning process in order to refine each others model
 - Agents' private data is never revealed: communication about individual training instance is performed in terms of their *ids*
 - At the end of the process, agents vote to decide the group's prediction
- Santana and colleagues (2006): **agent-based neural network classifier**; partially explore diversity among agents (agents have the same basic functioning, but different method's parameter values)
 - confidence-based negotiation: the agent with the highest confidence classifies unlabeled instances

Agent-Based Classification

- Recamonde-Mendoza and Bazzan (2016): extend Modi and Shen's work (recall: collaborative learning for distributed classification tasks) to **explore variation of algorithms among agents** (deal with heterogeneous agents); suitable for vertical partition
- Bazzan (2013): **classification is combined with RL** (states of the MDP are values of the attributes of the new instance to be classified; actions are the possible values of the target class)
- Wardeh *et al.* (2012): **classification combined with argumentation**: a group of agents argue about the classification of a given case according to their experience as recorded in individual local data sets

Agent-Based Clustering

- Standard clustering methods rely on central data structures
- Multiple agents can deal with multiple clustering criteria and/or multiple aspects of the data
- Privacy: only high-level information derived from data analysis is shared among agents

Agent-Based Clustering

- Kargupta et al. (1997): PADMA to deal with **distributed data sources and hierarchical clustering**
- Silva *et al.* (2005): agent-oriented algorithm for **privacy-preserving clustering** applied to sensor networks
- Tozicka et al. (2007): generic framework to deal with interaction **heterogeneous and/or self-interested learners**
- Chaimontree *et al.* (2010): agents use **distinct clustering algorithm**: K-means, K-nearest neighbors and DBSCAN
- Chaimontree *et al.* (2012): two phases involving 1. **bidding** for data points, 2. **negotiation in which agents pass individual records to each other to improve the initial result**
- Babu *et al.* (2009): **Clustering combined with decision trees** where agents do feature selection and use domain knowledge to generate a decision tree

Agent-Based Clustering

- Agogino and Tumer (2006)
 - **Ensemble of clusters** without using the original data points that were used to generate these sets
 - Use of *difference utilities* (a.k.a. COIN or WLU; Tumer and Wolpert (2000)) to maximize the global utility of the clustering
 - Task of cluster ensemble is different from clustering
 - **combines multiple clusterings**, formed from different aspects of the same data set, into a single unified clustering
 - goal is to create a single clustering that best characterizes a set of clustering, without using the original data points
 - Preserves privacy

Agent-Based Clustering

- **Multi-objective clustering**: one of the most interesting scenarios for using agents because it may involve competition, conflicts
 - dos Santos *et al.* (2009): MACC (multi ant colony clustering) inspired by ant colony optimization and multi-objective clustering
 - Simultaneously uses several ant colonies, each one aiming to optimize one particular objective
 - Find different shapes and sizes of clusters and different types of structures in a data set

Main Characteristics

- Majority of works aim at helping agents to predict behaviors of other agents or characteristics of the environment in which the agent acts
- Use of ML techniques is more diverse compared to the case of agent-based ML
- ML techniques frequently appear in combination with other AI techniques

Various techniques used

- Nunes and Oliveira (2004):
 - heterogeneous learning agents improving their learning skills by communicating with members of other groups that are solving similar problems in different areas
 - Agents use various ML techniques (Q-learning, neural networks) as well as heuristics
 - Application: several sources during learning is used in a simplified traffic control problem
- Fiosins et al. (2012): application of bootstrap-based CUSUM in decentralized routing of vehicles
- Urban *et al.* (2010): expectation-maximization to build a Gaussian mixture model and train a HMM to detect anomalous trajectories

Various techniques used

- Şensoy et al. (2013): **regression to discover patterns of interactions among agents** in order to estimate trust
- Emele et al. (2012): **decision trees extended with ontological reasoning, argumentation**, and use of domain knowledge to support learning of policies
- Cetnarowicz *et al.* (1996): **evolutionary approach for the task prediction** to improve the performance of agents

Clustering combined with other techniques

- Pan *et al.* (2013): clustering combined with KNN for labeling items in collaborative tagging and recommender systems
- Bekkerman et al. (2007): multiagent heuristic web searching algorithms for web page clustering agents
- Brahmi *et al.* (2012): association rules and clustering used for dealing with intrusion detection, where agents collect and analyze the network connections patterns
- Garruzzo and Rosaci (2008): clustering of agents based on similarity value that has lexical, structural and semantic components

Clustering combined with other techniques

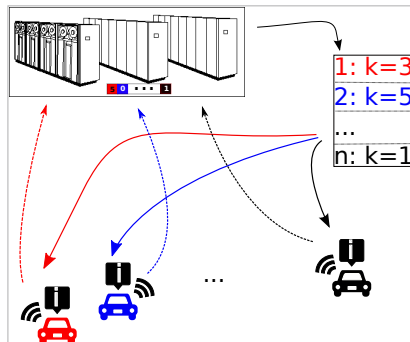
- Ogston *et al.* (2003): clustering of agents with similar objectives or data
- Piraveenan et al. (2008): predictor for convergence time of cluster formation in sensor grids
- dos Santos and Bazzan (2012): swarm intelligence, bee dance metaphor to cluster agents in the domain of the RoboCup Rescue

RL combined with GA

- Many real-world problems: conflict between desired performance of the system as a whole, and performance of individual components
- Example of congestion games: while a central authority is interested in optimizing the *average* travel time, drivers are interested in optimizing their own individual travel times
- Proposed solution: synergy between GA and RL
 - GA optimizes for the central authority (find an assignment of routes, one for each driver agent)
 - RL is used by driver agents to find a route that minimizes its travel time

RL combined with GA (cont.)

- Example of congestion games: while a central authority is interested in optimizing the *average* travel time, drivers are interested in optimizing their own individual travel times
- Proposed solution: synergy between GA and RL (Bazzan, 2018)



Transition from single agent to multiagent

- Advantages...
- ... but also challenges:
 - Modal logics, mental states
 - Distributed problem solving
 - Decision making in complex, dynamic, and partially observable environments
 - Multiagent learning
 - Learning / adaptation
 - Self organization and/or task allocation
 - Mix of techniques
 - Synergy between various techniques

Thank You !

- Acknowledgments
 - CNPq
 - Alexander von Humboldt Foundation
- Next slides contain references cited
- You find these slides at `www.inf.ufrgs.br/~bazzan`

- Agogino, A. and Tumer, K. (2006). Efficient agent-based cluster ensembles. In P. Stone and G. Weiss, editors, *AAMAS '06: Proceedings of the 5th International Joint Conference on Autonomous agents and Multiagent Systems*, pages 1079–1086, New York, NY, USA. ACM.
- Alam, S., Dobbie, G., and Riddle, P. (2009). Exploiting swarm behaviour of simple agents for clustering web users' session data. In L. Cao, editor, *Data Mining and Multi-agent Integration*, pages 61–75. Springer US.
- Babu, T. R., Murty, M. N., and Subrahmanya, S. V. (2009). Multiagent systems for large data clustering. In L. Cao, editor, *Data Mining and Multi-agent Integration*, pages 219–238. Springer US.
- Bazzan, A. (2013). Cooperative induction of decision trees. In *2013 IEEE Symposium on Intelligent Agents (IA)*, pages 62–69.
- Bazzan, A. L. C. (2009). Agents and data mining in bioinformatics: Joining data gathering and automatic annotation with classification and distributed clustering. In L. Cao, editor, *Proc. of the Workshop on Agents and Data Mining Interaction*, number 5680 in Lecture Notes in Artificial Intelligence, pages 3–20, Berlin. Springer-Verlag.
- Bazzan, A. L. C. (2018). Accelerating the computation of solutions in resource allocation problems using an evolutionary approach and multiagent reinforcement learning. In K. Sim and P. Kaufmann, editors, *Applications of Evolutionary Computation (EvoApplications 2018)*, volume 10784 of *Lecture Notes in Computer Science*, pages 185–201, Parma, Italy. Springer.
- Bekkerman, R., Zilberstein, S., and Allan, J. (2007). Web page clustering using heuristic search in the web graph. In M. M. Veloso, editor, *IJCAI*, pages 2280–2285.
- Bianchi, R. and Bazzan, A. L. C. (2012). Combining independent and joint learning: a negotiation based approach. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 1395–1396, Valencia. IFAAMAS.
- Brahmi, I., Yahia, S., Aouadi, H., and Poncelet, P. (2012). Towards a multiagent-based distributed intrusion detection system using data mining approaches. In L. Cao, A. L. C. Bazzan, A. L. Symeonidis, V. I. Gorodetsky, G. Weiss, and P. S. Yu, editors, *Agents and Data Mining Interaction*, volume 7103 of *Lecture Notes in Computer Science*, pages 173–194. Springer Berlin Heidelberg.
- Brazdil, P. and Muggleton, S. (1991). Learning to relate terms in a multiple agent environment. In Y. Kodratoff, editor, *European Working Session on Learning (EWSL-91)*, volume 482 of *Lecture Notes in Computer Science*, pages 424–439. Springer Berlin Heidelberg.

- Brazdil, P., Gams, M., Sian, S. S., Torgo, L., and de Velde, W. V. (1991). Panel: Learning in distributed systems and multi-agent environments. In Y. Kodratoff, editor, *European Working Session on Learning (EWSL-91)*, volume 482 of *Lecture Notes in Computer Science*, pages 412–423. Springer.
- Cao, L., Luo, C., and Zhang, C. (2007). Agent-mining interaction: An emerging area. In V. Gorodetsky, C. Zhang, V. A. Skormin, and L. Cao, editors, *AIS-ADM*, volume 4476 of *Lecture Notes in Computer Science*, pages 60–73. Springer.
- Cetnarowicz, K., Kisiel-Dorohinicki, M., and Nawarecki, E. (1996). The application of evolution process in multi-agent world to the prediction system. In M. Tokoro, editor, *Proceedings of the International Conference on Multiagent Systems (ICMAS 96)*, pages 26–32. AAAI Press.
- Chaimontree, S., Atkinson, K., and Coenen, F. (2010). Clustering in a multi-agent data mining environment. In L. Cao, A. L. Bazzan, V. Gorodetsky, P. A. Mitkas, G. Weiss, and P. S. Yu, editors, *Agents and Data Mining Interaction*, volume 5980 of *Lecture Notes in Computer Science*, pages 103–114. Springer Berlin Heidelberg.
- Chaimontree, S., Atkinson, K., and Coenen, F. (2012). A framework for multi-agent based clustering. *Autonomous Agents and Multi-Agent Systems*, **25**(3), 425–446.
- Şensoy, M., Yilmaz, B., and Norman, T. J. (2013). Discovering frequent patterns to bootstrap trust. In L. Cao, Y. Zeng, A. L. Symeonidis, V. I. Gorodetsky, P. S. Yu, and M. P. Singh, editors, *Agents and Data Mining Interaction*, volume 7607 of *Lecture Notes in Computer Science*, pages 93–104. Springer Berlin Heidelberg.
- Davies, W. and Edwards, P. (1995). Distributed learning: An agent-based approach to data-mining. In *Proceedings of Machine Learning Workshop on Agents that Learn from Other Agents*.
- dos Santos, D. S. and Bazzan, A. L. C. (2012). Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach. *Appl. Soft Comput.*, **12**(8), 2123–2131.
- dos Santos, D. S., de Oliveira, D., and Bazzan, A. L. C. (2009). A multiagent, multiobjective clustering algorithm. In L. Cao, editor, *Data Mining and Multiagent Integration*. Springer.
- Emele, C. D., Norman, T. J., Şensoy, M., and Parsons, S. (2012). Learning strategies for task delegation in norm-governed environments. *Autonomous Agents and Multi-Agent Systems*, **25**(3), 499–525.
- Fiosins, M., Fiosina, J., and Müller, J. P. (2012). Change point analysis for intelligent agents in city traffic. In L. Cao, A. L. C. Bazzan, A. L. Symeonidis, V. I. Gorodetsky, G. Weiss, and P. S. Yu, editors, *Agents and Data Mining Interaction*, volume 7103 of *Lecture Notes in Computer Science*, pages 195–210. Springer Berlin Heidelberg.

- Garruzzo, S. and Rosaci, D. (2008). Agent clustering based on semantic negotiation. *ACM Transactions on Autonomous and Adaptive Systems*, 3(2), 7:1–7:40.
- Hindriks, K. V. and Tykhonov, D. (2008). Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 331–338.
- Kargupta, H., Kargupta, H., Hamzaoglu, I., Hamzaoglu, I., Stafford, B., and Stafford, B. (1997). Scalable, Distributed Data Mining Using an Agent Based Architecture. In *Proceedings of the Third International Conference on the Knowledge Discovery and Data Mining*, pages 211–214, Menlo Park, California, USA. AAAI Press.
- Kargupta, H., Park, B., Hershberger, D., and Johnson, E. (1999). Collective data mining: A new perspective toward distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery*, volume 2, pages 131–174. MIT Press, Cambridge, MA, USA.
- Kiekintveld, C., Miller, J., Jordan, P. R., and Wellman, M. P. (2007). Forecasting market prices in a supply chain game. In *6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1318–1325.
- Lemmens, N. and Tuyls, K. (2009). Stigmergic landmark foraging. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 497–504.
- Modi, P. J. and Shen, W.-M. (2001). Collaborative Multiagent Learning for Classification Tasks. In *Proceedings of the 5th International Conference on Autonomous Agents*, AGENTS '01, pages 37–38, New York, NY, USA. ACM.
- Nunes, L. and Oliveira, E. C. (2004). Learning from multiple sources. In N. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi Agent Systems*, AAMAS, volume 3, pages 1106–1113, New York, USA. New York, IEEE Computer Society.
- Ogston, E., Overeinder, B., van Steen, M., and Brazier, F. (2003). A method for decentralized clustering in large multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS '03), pages 789–796, New York, NY, USA. ACM.
- Oliveira, D. de. and Bazzan, A. L. C. (2009). Multiagent learning on traffic lights control: effects of using shared information. In A. L. C. Bazzan and F. Klügl, editors, *Multi-Agent Systems for Traffic and Transportation*, pages 307–321. IGI Global, Hershey, PA.
- Pan, R., Dolog, P., and Xu, G. (2013). KNN-based clustering for improving social recommender systems. In L. Cao, Y. Zeng, A. L. Symeonidis, V. I. Gorodetsky, P. S. Yu, and M. P. Singh, editors, *Agents and Data Mining Interaction*, volume 7607 of *Lecture Notes in Computer Science*, pages 115–125. Springer Berlin Heidelberg.

- Piraveenan, M., Prokopenko, M., Wang, P., and Zeman, A. (2008). Decentralized multi-agent clustering in scale-free sensor networks. In J. Fulcher and L. C. Jain, editors, *Computational Intelligence: A Compendium*, volume 115 of *Studies in Computational Intelligence*, pages 485–515. Springer.
- Recamonde-Mendoza, M. and Bazzan, A. L. (2016). Social choice in distributed classification tasks: Dealing with vertically partitioned data. *Information Sciences*, **332**(1), 56–71.
- Santana, L. E. A., Canuto, A. M. P., Xavier, Jr., J. C., and Campos, A. M. C. (2006). A Comparative Analysis of Data Distribution Methods in an Agent-Based Neural System for Classification Tasks. In *HIS*, page 9.
- Sian, S. (1991). Adaptation based on cooperative learning in multi-agent systems. In Y. Demazeau and J. Müller, editors, *Decentralized A.I.*, volume 2, pages 257–272. North-Holland.
- Silva, J. C. d., Giannella, C., Bhargava, R., Kargupta, H., and Klusch, M. (2005). Distributed data mining and agents. *Engineering Applications of Artificial Intelligence*, **18**(7), 791–807.
- Stolfo, S., Tselepis, A. L. P. S., Prodromidis, A. L., Tselepis, S., Lee, W., Fan, D. W., and Chan, P. K. (1997). JAM: Java agents for meta-learning over distributed databases. In *Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining*, pages 74–81. AAAI Press.
- Stone, P. (2007). Learning and multiagent reasoning for autonomous agents. In *The 20th International Joint Conference on Artificial Intelligence*, pages 13–30.
- Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, **8**(3), 345–383.
- Tozicka, J., Rovatsos, M., and Pechoucek, M. (2007). A framework for agent-based distributed machine learning and data mining. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '07)*, pages 666–673, New York, NY, USA. ACM.
- Tumer, K. and Wolpert, D. (2000). Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109. AAAI Press.
- Tuyls, K. and Weiss, G. (2012). Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, **33**(3), 41–52.
- Urban, S., Jakob, M., and Pěchouček, M. (2010). Probabilistic modeling of mobile agents' trajectories. In L. Cao, A. L. C. Bazzan, V. Gorodetsky, P. A. Mitkas, G. Weiss, and P. S. Yu, editors, *Agents and Data Mining Interaction*, volume 5980 of *Lecture Notes in Computer Science*, pages 59–70. Springer Berlin Heidelberg.
- Wardeh, M., Coenen, F., and Bench-Capon, T. (2012). Multi-agent based classification using argumentation from experience. *Autonomous Agents and Multi-Agent Systems*, **25**(3), 447–474.

- Zhang, W. and Wang, Y. (2012). Agent-based cluster analysis of tropical cyclone tracks in the western north pacific. In L. Cao, A. L. C. Bazzan, A. L. Symeonidis, V. I. Gorodetsky, G. Weiss, and P. S. Yu, editors, *Agents and Data Mining Interaction*, volume 7103 of *Lecture Notes in Computer Science*, pages 98–113. Springer Berlin Heidelberg.