Formulae

General

- ***Subgame Perfect Equilibria***
  Remember it is a subset of the NE of the game, so just look at them and check if they would be senseful choices.
  If you have a tree representation I'm positive you can find one of them (maybe that's it) by doing backtracking on the tree.
- ***Nash Equilibria*** can be found by doing the intersection between the best response functions

Markov Decision Processes

Definition

- S
- A
- $P : S \times S \times A \to [0, 1]$
- $R : S \times A \to \mathbb{R}$
- λ

Bellman Equations

Bellman Expectation Equation

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \left[R(s, \pi(s)) + \lambda \sum_{s' \in S} P\big(s'|s, \pi(s)\big) V^{\pi}(s') \right]$$

Bellman Optimality Equation

$$V^{*}(s) = \max_{a \in A} \left\{ R(s, a) + \lambda \sum_{s' \in S} P(s'|s, a) V^{*}(s') \right\}$$

alternative version (in case R is defined as $R : S \to \mathbb{R}$) :

$$V^{*}(s) = R(s) + \lambda \max_{a \in A} \left\{ \sum_{s' \in S} P(s'|s, a) V^{*}(s') \right\}$$

Value Iteration

Iteratively find a better value function until convergence.

(S, A, P, R, λ) returns $\hat{V}^*$ m that is an approximation of V^*

Algorithm

---

initialize $\hat{V}^*$

repeat

$V \leftarrow \hat{V}^*$

for each $s \in S$ do

$\hat{V}^*(s) \leftarrow \max_{a \in A} \left[R(s,a) + \lambda \sum_{s' \in S} P(s'|s,a) V(s') \right]$

end for

until $\max_{s \in S} |V(s) - \hat{V}^*(s)| < \varepsilon$

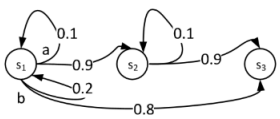
Be careful: i think Amigoni prefers the alternative version of the update (see *Bellman Optimality Equation*).

Here I'm not looking for something that is in my equation, I have an estimate of V!

Exercises

First

Question 1 (8 points). Consider the following **Markov Decision Process**:



Find the interval of values for discount factor γ such that the agent strictly prefers action *a* to action *b* in s_1. Assume that $r(s_1)=r(s_2)=0$ and $r(s_3)=1$.
Explain the results intuitively.

The agent prefers action *a* to action *b* in s_1 when:

(*) $u(s_1)\cdot 0.1 + u(s_2)\cdot 0.9 > u(s_1)\cdot 0.2 + u(s_3)\cdot 0.8$

Utilities of states can be calculated solving:

$u(s_1) = 0 + \gamma \max\{u(s_1)\cdot 0.2 + u(s_3)\cdot 0.8, u(s_1)\cdot 0.1 + u(s_2)\cdot 0.9\}$

$u(s_2) = 0 + \gamma(u(s_2)\cdot 0.1 + u(s_3)\cdot 0.9)$

$u(s_3) = 1$

From the last two equations we have:

$u(s_2) = \frac{9}{10\cdot(1-0.1\gamma)}\gamma$

The first equation becomes:

$u(s_1) = \gamma \max\left\{u(s_1)\cdot 0.2 + 0.8, u(s_1)\cdot 0.1 + \frac{9}{10\cdot(1-0.1\gamma)}\gamma\cdot 0.9\right\}$

We exploit preference of action *a* over action *b*:

$u(s_1) = \gamma u(s_1)\cdot 0.1 + \frac{9}{10\cdot(1-0.1\gamma)}\gamma^2\cdot 0.9$

and get:

$u(s_1) = \frac{9}{100}\cdot\frac{9}{(1-0.1\gamma)^2}\gamma^2$

Finally, inequality (*) is verified when:

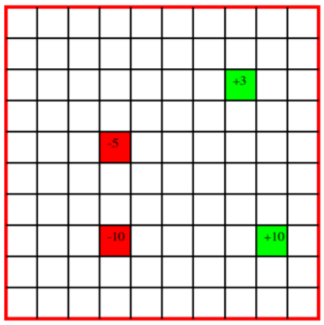
$\frac{9}{100}\cdot\frac{9}{(1-0.1\gamma)^2}\gamma^2\cdot 0.1 + \frac{9}{10\cdot(1-0.1\gamma)}\gamma\cdot 0.9 > \frac{9}{100}\cdot\frac{9}{(1-0.1\gamma)^2}\gamma^2\cdot 0.2 + 0.8$

and, after some math, we discover that it is never satisfied for any value of $\gamma\in[0,1]$ (only equality h for $\gamma=1$).

Intuitively, this means that the agent will never prefer action *a* to action *b*, independently of discount applied to future rewards.

Second

Question 1 (8 points). Consider the following 10x10 grid environment.



A robot operates in this environment. When in a cell, it can choose one of four actions: up, down, left right. When the robot selects one of these actions, it has a 0.7 chance of going one step in the desi direction and 0.1 chance of going in any of the other three directions. If it bumps into the outside w the agent does not actually move. There are four rewarding cells, as shown in the figure. The rewar the other cells is 0.
Suppose that the discount factor is γ=0.9 and that $u^{t=0}(s)$=0 for all states *s*.
Using the **value iteration** algorithm calculate the values of the 9 cells around the cell with +10 rew after the first and the second iteration of the algorithm (t=1 and t=2).
According to the values at t=2, what is the optimal policy for the robot in the cell immediately on left of the cell with +10 reward?

The value iteration algorithm updates the values of cells (states) *s* according to:

$$u^{t+1}(s) \leftarrow r(s) + \gamma \max_a \sum_{s'} T(s,a,s') \cdot u^t(s)$$

The values are reported in the following.

0	0	0
0	10	0
0	0	0

0	6.3	0
6.3	10	6.3
0	6.3	0

According to the values at t=2, the optimal policy for the robot in the cell immediately on the left of cell with +10 reward is to go to the right (all other actions give 0 to the robot).

Negotiation

Definitions

Alternatives

$$A = \{z|\text{condition on } z\} \cup D$$

where D is the disagreement

Bargaining Set

$$S = \{(U^a(z), U^b(z))|z \in A\}$$

Disagreement Point

$$d = (U^a(D), U^b(D)) = (d^a, d^b)$$

Bargaining Problem

$$(S, d)$$

Bargaining Solution

$$f : \{(S, d)\} \to S$$

Nash Bargaining Solution

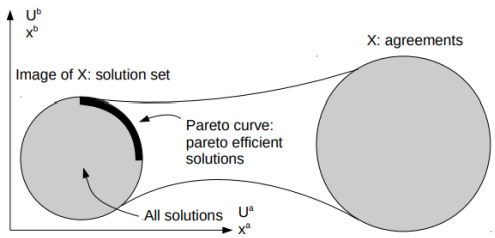
$$f(S,d) = \underset{x=(x^a,x^b)\in S \text{ and } x\geq d}{\text{argmax}} (x^a - d^a)(x^b - d^b)$$

Solutions' properties

1. Pareto efficiency

x is pareto dominated by x' if
$\forall i \in N | U_i(x') \geq U_i(x)$,
$\exists i \in N | U_i(x') > U_i(x)$

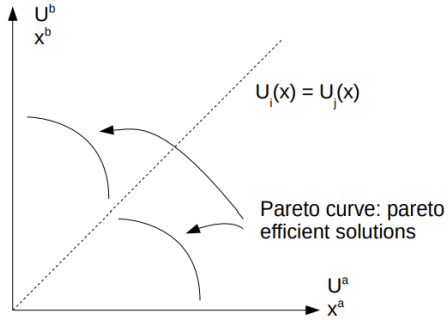
x is pareto efficient if
$\nexists x' | x$ *is* pareto dominated by x'



at least always one solution

2. Egalitarian solution

$U_i(x) = U_j(x) \forall i, j$ and $\max_x U_i(x)$



could not exist

3. Utilitarian solution

$x = \underset{x}{argmax} \underbrace{\sum_i U_i(x)}_{\text{social welfare}}$ always exists and pareto efficient (particular point in pareto curve)

4. Nash Bargaining solution

$x = \underset{x}{argmax} \prod_i U_i(x)$ always exists and pareto efficient (particular point in pareto curve)

$\bar{x} = [x_1, \ldots, x_n]$

Pareto Efficiency

if $n = 2$:

1. plot $(U_1(\bar{x}), U_2(\bar{x}))$
2. if the point corresponding to such pair of utilities doesn't have other pairs of utilities above or on the right, it is a Pareto Efficient solution.

Egalitarian

$\bar{x} = [x_1, \ldots, x_n]$

$\forall i, j \quad U_i(\bar{x}) = U_j(\bar{x})$

$\max U_i(\bar{x})$

Utilitarian

(maybe, I could refer to the solution that satisfy this property also as social welfare solutions... not sure!)

$\bar{x} = [x_1, \ldots, x_n]$

$x^* = \underset{\bar{x}}{\operatorname{argmax}} \left(\sum_i U_i(\bar{x}) \right)$

Nash Bargaining Solution

$\bar{x} = \underset{\bar{x}}{\operatorname{argmax}} \left(\prod_i U_i(x) \right)$

Other types of solution (I think they have not been mentioned in class)

- Egalitarian Social Welfare - guarantees pareto optimal solution
- Kalai-Smorodinski - does not guarantee p.o. solution, but yes if the set of deals is continuous

Rubinstein's Alternating Offer Protocol

- two agents, a and b
- a good that can be shared, with shares $x^a + x^b = 1$
- discrete time $t = 1, 2, \dots$
- discount factors δ^a, δ^b with $0 \leq \delta^a, \delta^b \leq 1$

Utilities

$$U^a(x_a, t) = x^a \cdot \delta^{t-1}$$

$$U^b(x_b, t) = x^b \cdot \delta^{t-1}$$

Equilibrium Shares

$$x^a = \frac{1 - \delta^b}{1 - \delta^a \delta^b}$$

$$x^b = \frac{\delta_b - \delta_a \delta_b}{1 - \delta_a \delta_b}$$

Monotonic Concession Protocol

General

- agents a, b ;
- utilities U^a, U^b
- simultaneous offer protocol. the offer is usually referred to as x
- $t = 1, 2, \dots$

Algorithm

From the point of view of agent a:

1. $x^{(a)} \leftarrow \arg\max_{x \in O} U^a(x)$

2. propose $x^{(a)}$

3. receive $x^{(b)}$

4. if $U^a(x^{(b)}) \geq U^a(x^{(a)})$ then ACCEPT $x^{(b)}$ and RETURN
   else $x^{(a)} \leftarrow \underset{x \in (O \setminus x^a)}{\operatorname{argmax}} U^b(x)$ such that $U^b(x) \geq U^b(x^{(a)})$ and $U^a(x) \geq 0$

5. goto 2

The green part is usually re-written as follows: $U^b_{t+1}(x) = U^b_t(x) + \varepsilon$, with ε being a specified constant.

Exercise Example

Question 3 (8 points). Consider a **bargaining** scenario with two agents negotiating over the interval real-valued issues $x \in [1,4]$ with the following utility functions $u_1(x)=16-x^2$ and $u_2(x)=2+x^2$.
1. Apply the **monotonic concession protocol** to find an agreement, assuming concession $\varepsilon=3$.
2. Which is the first agent to concede when using the **Zeuthen strategy**? Why?
3. Calculate the **Nash bargaining solution** for the above bargaining scenario.

1. The evolution of the negotiation using the monotonic concession protocol is as follows:

	agent 1	**agent 2**
step 0	$\delta_1^{t=0}=1$	$\delta_2^{t=0}=4$
	exchange proposals	
	$u_1(\delta_1^{t=0})=15>0=u_1(\delta_2^{t=0})$ agent 1 concedes	$u_2(\delta_2^{t=0})=18>3=u_2(\delta_1^{t=0})$ agent 2 concedes
step 1	$\delta_1^{t=1}$ such that $u_2(\delta_1^{t=1})=u_2(\delta_1^{t=0})+\varepsilon$ $2+(\delta_1^{t=1})^2=3+3$ $\delta_1^{t=1}=2$	$\delta_2^{t=1}$ such that $u_1(\delta_2^{t=1})=u_1(\delta_2^{t=0})+\varepsilon$ $16-(\delta_2^{t=1})^2=0+3$ $\delta_2^{t=1}=\sqrt{13}$
	exchange proposals	
	$u_1(\delta_1^{t=1})=12>3=u_1(\delta_2^{t=1})$ agent 1 concedes	$u_2(\delta_2^{t=1})=15>6=u_2(\delta_1^{t=1})$ agent 2 concedes
step 2	$\delta_1^{t=2}$ such that $u_2(\delta_1^{t=2})=u_2(\delta_1^{t=1})+\varepsilon$ $2+(\delta_1^{t=2})^2=6+3$ $\delta_1^{t=2}=\sqrt{7}$	$\delta_2^{t=2}$ such that $u_1(\delta_2^{t=2})=u_1(\delta_2^{t=1})+\varepsilon$ $16-(\delta_2^{t=2})^2=3+3$ $\delta_2^{t=2}=\sqrt{10}$
	exchange proposals	
	$u_1(\delta_1^{t=2})=9>6=u_1(\delta_2^{t=2})$ agent 1 concedes	$u_2(\delta_2^{t=2})=12>9=u_2(\delta_1^{t=2})$ agent 2 concedes
step 3	$\delta_1^{t=3}$ such that $u_2(\delta_1^{t=3})=u_2(\delta_1^{t=2})+\varepsilon$ $2+(\delta_1^{t=3})^2=9+3$ $\delta_1^{t=3}=\sqrt{10}$	$\delta_2^{t=3}$ such that $u_1(\delta_2^{t=3})=u_1(\delta_2^{t=2})+\varepsilon$ $16-(\delta_2^{t=3})^2=6+3$ $\delta_2^{t=3}=\sqrt{7}$
	exchange proposals	
	$u_1(\delta_1^{t=3})=6<9=u_1(\delta_2^{t=3})$ agent 1 accepts	$u_2(\delta_2^{t=3})=9>12=u_2(\delta_1^{t=3})$ agent 2 accepts

2. The first offers of the two agents are, as before: $\delta_1^{t=0}=1$ and $\delta_2^{t=0}=4$. Both agents calculate their risks:

$risk_1=\frac{u_1(\delta_1^{t=0})-u_1(\delta_2^{t=0})}{u_1(\delta_1^{t=0})}=\frac{15-0}{15}=1$ and $risk_2=\frac{u_2(\delta_2^{t=0})-u_2(\delta_1^{t=0})}{u_2(\delta_2^{t=0})}=\frac{18-3}{18}=\frac{5}{6}$

Since its risk is smaller, agent 2 concedes first.

3. The Nash bargaining solution $\bar{x}$ is such that $\bar{x}=\arg\max_{x\in[1,4]} u_1(x)\cdot u_2(x)$, namely it maximizes (let call $x^2=z$): $(16-z)\cdot(2+z)=-z^2+14z+32$.

Finding the maximum: $-2z+14=0$ and $z=7$, which means $\bar{x}=\sqrt{7}$.

Zeuthen Strategy

General

The idea is that you have a high risk when you are close to the situation in which your utility is very small, when you are close to the no-agreement. The agent that has the smallest risk will concede and make the next offer.
Property: the final agreement found via this algorithm is a pareto optimal agreement, which means that there is no another agreement that is not worse for one agent and strictly better for the other one.

It finds an agreements which is a Nash BARGAINING solution. It is an agreement that maximizes the product of the two utilities.

Algorithm

This is the point of view of a

1. $x^{(a)} \leftarrow \arg\max_{x \in O} U^a(x)$

2. propose $x^{(a)}$

3. receive $x^{(b)}$

4. if $U^a(x^{(b)}) \geq U^a(x^{(a)})$ then ACCEPT $x^{(b)}$ and RETURN

5. $\text{risk}_a \leftarrow \frac{U^a(x^{(a)}) - U^a(x^{(b)})}{U^a(x^{(a)})}$; $\text{risk}_b \leftarrow \frac{U^b(x^{(b)}) - U^b(x^{(a)})}{U^b(x^{(b)})}$

6. if $\text{risk}_a < \text{risk}_b$ then $x^{(a)} \leftarrow x \in O$ such that $\text{risk}_a > \text{risk}_b$

7. goto 2

The green part is explicated as follows (suppose agent 1 is the one that needs to come up with the new offer)

$$\frac{U^1(x_{t+1}^{(a)}) - U^1(x_t^{(b)})}{U^1(x_{t+1}^{(a)})} \geq \frac{U^2(x_t^{(b)}) - U^2(x_{t+1}^{(a)})}{U^2(x_t^{(b)})}$$

in short you need:

- the utility functions of both agents
- the utility of both agents with the old offer of the agent that is waiting for the new offer

Computational Social Choice

Voting Rules and Definitions

Disclaimers

- voting rules means scoring rules
- Strict Majority Voting differs from Plurality Voting!

Plurality Voting

One point for each vote, everyone votes for his preferred alternative. The winner is the alternative with the highest score.

Borda Count

Consider the preference relation expressed by each agent.
Suppose you have m alternatives. weight first alternative $m-1$, second alternative $m-2$, and so on.

Pile wise majority voting

a voting where you have only 2 alternatives and the alternative that gets more points is the winner

Condorcet Winner

An alternative that wins against every other alternative in a pile wise majority voting

Condorcet Extension

Voting rule that selects a Condorcet winner if a Condorcet winner exists.

- Sequential Majority voting *is* a condorcet extension
- Borda Count *is NOT* a condorcet extension

Copeland's Rule

In order to determine the winner you give to each alternative one point every time it is winning against another alternative, you give $1/2$ when there is a tie, 0 points when it loses.

Plurality with Runoff

In the first stage you run a plurality voting.
Then you take the 2 most voted candidates and run a pairwise majority voting between the 2 (a ballot).

Example

6	**5**	**4**
milk	beer	wine
wine	wine	beer
beer	milk	milk

after 1^{st} stage:

milk	**beer**
6	5

after 2^{nd} stage:

beer wins over milk (9 over 6):

Conclusion: Beer wins.

Single Transferable Vote (SVT)

Run a plurality voting, then the alternatives that are ranked first by the lowest number of agents are eliminated, then you run a new plurality voting without considering the eliminated candidates.
N candidates $\rightarrow N-1$ round to perform.

Example

6	**5**	**4**
milk	beer	wine
wine	wine	beer
beer	milk	milk

I eliminate wine.

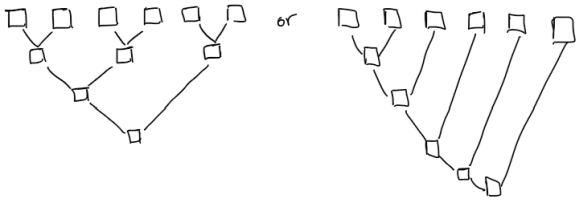
6	**5**	**4**
milk	beer	beer
beer	milk	milk

beer wins over milk (9 over 6):

I eliminate milk. Beer wins. Is it a condorcet winner extension? no.

Sequential Majority Voting

Combine in different ways a sequence of majority voting.



Sequential Majority Voting is a condorcet extension because the condorcet winner can never lose in a pair wise comparison.

Gibbard–Satterthwaite Theorem

Any voting system is prone to strategic manipulation.

So you can not avoid strategic manipulation, but you can make it harder to happen.

Another Gibbard-Satterthwaite Theorem

when

- *there are more than 2 alternatives*
- *you use a probabilistic voting rule*
- *you impose that your probabilistic voting system is Pareto optimal*
- *and it is no manipulable*

then the only probabilistic voting system that is satisfying it is probabilistic dictatorship.

Example on Condorcet Winner

Question 4 (7 points). Consider a **voting** setting. (1) Define the Condorcet winner. (2) Does the **sequential majority** (with any agenda) and the **Borda count** satisfy the Condorcet winner condition? Explain why. A *Condorcet loser* is a candidate that loses against every other candidate in a pairwise election. (3) Can the **sequential majority** (with any agenda) and the **Borda count** choose a Condorcet loser as winner? Explain why considering a generic voting setting with three agents (1, 2, 3) and three candidates (a, b, c).
(1) A Condorcet winner is a candidate that wins against every other candidate in a pairwise election. (2) The sequential majority satisfies the Condorcet winner condition because a Condercet winner will trivially wins all the elections. The Borda count does not satisfy the Condorcet winner condition, as this counter-example shows: 5 agents: a > b > c 4 agents: b > c > a Where candidate a is the Condorcet winner, but candidate b wins the election with Borda count (a gets 19 points, b 22 points, and c 13 points). (3) The sequential majority cannot choose a Condorcet loser by definition because that candidate will lose the first election it takes part in. Also the Borda count cannot choose a Condorcet loser. For example, with three agents (1, 2, 3) and three candidates (a, b, c) , in order for a candidate (let's say c) to be a Condorcet loser, it should be ranked last by at least two agents (let's say 1 and 2): agent 1: a > b > c agent 2: b > a > c In the best case, c is ranked first by the third agent: agent 3: c > a > b But it collects 1+1+3=5 points that are not enough for winning the election using Borda count.

Auctions

Single Item Canonical Auctions

General Disclaimer

- the goal of the agents is not to get the item paying their private value, but to maximize their utility

English Auctions
The auctioneer announces an initial price (the reservation price).
The agent can make a new offer with the following constraint: the new offer should be larger than the largest offer made so far by any of the other agents.
When the auction in this case ends? when there is no new offer.
The agent will pay exactly the amount of money that he has offered.

- offer made by agents
- usually big jump of prices
- the auctioneer is more passive. he has no control
- the winning strategy is to offer an increase until my true evaluation

Japanese Auctions
At the beginning all the agents will be standing up. The auctioneer will start from his reservation price and he will call increasing prices.
If an agent sits down he is out of the auction forever.
The winner is the last standing agent.
The amount of money he will pay is the price that the auctioneer has proposed at last.
What happens if we have two agents standing and they sit down at the same time?
One tie-breaking mechanism is: a new auction is run just for those agents.

- offer made by auctioneer.
- more linear.
- the auctioneer is more in control of the situation.
- the winning strategy is to stay up until the auctioneer reaches your evaluation

Dutch Auctions
there is a clock. it does not show the time but the price.
it starts from a really high price and goes down.
the auction ends when one of the agents stops the clock.
The winner is the agent that stopped the clock and the amount of money he will pay is the price that is displayed by the clock at time of stopping.
(i.e. for selling flowers, for selling fish. it is really fast. used for selling things that are not of a big value and you have to sell them fast.
the auctioneer can set the speed of the clock)

- there is no dominant strategy

Sealed-bid Auctions

the agents send their bids (offer) to the auctioneer using a sealed envelope.
There are several families of sealed-bid auctions, but we'll see just two:

First Price Sealed-bid Auctions

The auctioneer selects the largest possible offer.
The agent that made the best offer will get the item and pays exactly the amount he has offered.

- Dutch and FPSB are strategically equivalent
- there is no dominant strategy in this case

Second Price Sealed-bid Auctions (also called Vickrey auctions)

The auctioneer selects the largest possible offer.
the winner is the one that made the largest offer but he does not pay his offer, but the second highest.

- offering your true evaluation of the object is the dominant strategy

Example

Question 4 (8 points). Consider an auction over a single item with private value that involves 4 bidding agents (1, 2, 3, and 4) that evaluate the item r_1=10, r_2=30, r_3=20, and r_4=15, respectively (in this case evaluations coincide with the reservation prices). 1) Assume that agents do not know the evaluations of other agents. Given the available information, is it possible to predict which agent will win and how much will it pay when using **English**, **Dutch**, **first-price sealed-bid**, and **Vickrey** auction? 2) Assume that agent 2 (and only it) knows the evaluations of other agents. How should agent 2 play in **English**, **Dutch**, **first-price sealed-bid**, and **Vickrey** auction?
1) English auction: the winner will be agent 2, paying 20 or something more than 20. Dutch auction: the winner cannot be predicted without knowing the strategies of the agents. First-price sealed-bid auction: the winner cannot be predicted without knowing the strategies of the agents. Vickrey auction: the winner will be agent 2 and it will pay exactly 20 (the second-best offer). 2) English auction: if possible according to the rules of the auction, agent 2 should immediately offer 20, resulting in agent 2 winning and paying 20. Otherwise, agent 2 will win the object paying 20 or something more than 20, as in the previous case. Dutch auction: agent 2 should wait until the price is slightly larger than 20 and then stop the auction, resulting in agent 2 winning and paying something more than 20. First-price sealed-bid auction: agent 2 should bid a price slightly larger than 20, resulting in agent 2 winning and paying something more than 20. Vickrey auction: agent 2 should offer 30 as in the previous case, resulting in agent 2 winning and paying exactly 20 (the second-best offer).

Coalition Formation

Superadditivity

$$v(\{C \cup D\}) \geq v(C) + v(D) \quad C \text{ and } D \text{ disjoint}$$

Convex Game

$$v(C \cup D) + v(C \cap D) \geq v(C) + v(D) \;\; \forall C, D \subseteq A$$

if a game is convex then it is even superadditive.

for convex games, the payoff vector composed of Shapley Values is always in the core (is always stable).

Simple Game

$$v(C) \in \{0, 1\} \; \forall C \in A$$

Shapley Value

It embeds the idea of fairness.

$$\varphi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! \, (n - |S| - 1)!}{n!} (v(S \cup \{i\}) - v(S))$$

Core

It embeds the idea of stability.
Compute the inequality for all coalitions but the grand coalition, and find the values of the payoff vector such that all of them are satisfied.

$$\text{core} = \{(CS, \bar{x}) | \sum_{i \in C} x_i \geq v(C) \;\; \forall C \subseteq A\}$$

Best Coalition Structure

Dynamic Programming Algorithm

The characteristic function of the game is:

S	{1}	{2}	{3}	{1, 2}	{1, 3}	{2, 3}	{1, 2, 3}
$v(S)$	10	10	0	25	5	5	20

(1) The game is not superadditive, since for example $v(\{1,2,3\}) = 20 < 25+0 = v(\{1,2\}) + v(\{3\})$.

(2) The computation of the DP algorithm proceeds as follows:

Coalition	**Evaluations**	***f*()**	***t***
{1}	$v(\{1\}) = 10$	10	{1}
{2}	$v(\{2\}) = 10$	10	{2}
{3}	$v(\{3\}) = 0$	0	{3}
{1,2}	$v(\{1,2\}) = 25$ $f(\{1\}) + f(\{2\}) = 20$	25	{1,2}
{1,3}	$v(\{1,3\}) = 5$ $f(\{1\}) + f(\{3\}) = 10$	10	{1},{3}
{2,3}	$v(\{2,3\}) = 5$ $f(\{2\}) + f(\{3\}) = 10$	10	{2},{3}
{1,2,3}	$v(\{1,2,3\}) = 20$ $f(\{1,2\}) + f(\{3\}) = 25$ $f(\{1,3\}) + f(\{2\}) = 20$ $f(\{2,3\}) + f(\{1\}) = 20$	25	{1,2},{3}

The optimal coalition structure is thus: {1,2}, {3}.

Shehory and Kraus

By the point o view of agent i.

(1) $C_i \leftarrow$ coalitions that include a_i

(2) $C_i^* \leftarrow \arg\max_{C \in C_i} \frac{v(C)}{|C|}$

(3) $\text{broadcast}(a_i, C_i^*), \text{receive}(a_j, C_j^*)$

(4) $C_{max} \leftarrow$ largest set of agents such that, for all $a_j \in C_{max}, (a_j, C_{max})$

(5) if $a_i \in C_{max}$ then join C_{max} and return

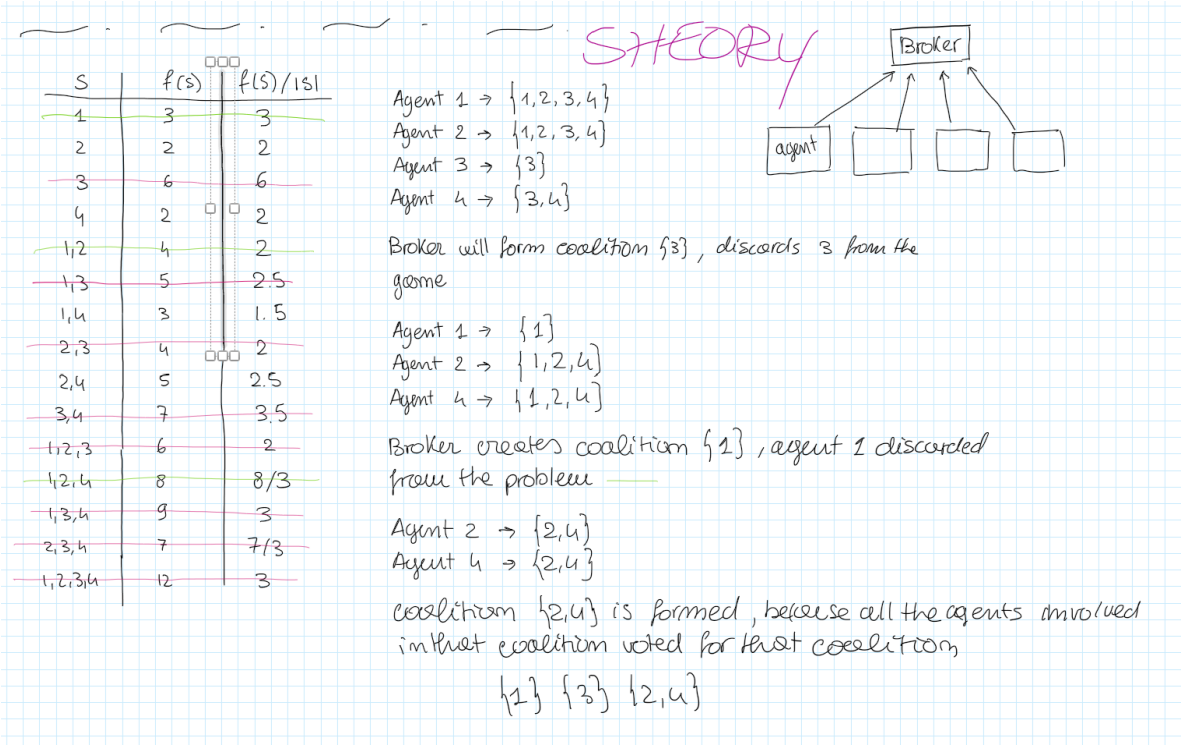
(6) delete from C_i coalitions that contain agents in C_{max}

(7) goto (2)

This algorithm is performed by all agents in parallel knowing the same information.

Examples

First



S	f(S)	f(S)/|S|
1	3	3
2	2	2
3	6	6
4	2	2
1,2	4	2
1,3	5	2.5
1,4	3	1.5
2,3	4	2
2,4	5	2.5
3,4	7	3.5
1,2,3	6	2
1,2,4	8	8/3
1,3,4	9	3
2,3,4	7	7/3
1,2,3,4	12	3

Agent 1 → {1,2,3,4}
Agent 2 → {1,2,3,4}
Agent 3 → {3}
Agent 4 → {3,4}

Broker will form coalition {3}, discards 3 from the game

Agent 1 → {1}
Agent 2 → {1,2,4}
Agent 4 → {1,2,4}

Broker creates coalition {1}, agent 1 discarded from the problem

Agent 2 → {2,4}
Agent 4 → {2,4}

coalition {2,4} is formed, because all the agents involved in that coalition voted for that coalition

{1} {3} {2,4}

Second

Question 3 (8 points). Consider a **coalitional game** with four agents Ag={1,2,3,4} and the following characteristic function:

$$v(C)=\begin{cases}|C|^2 & \text{if } |C|\leq 3\\ 0 & \text{otherwise}\end{cases}$$

1. Is the coalitional game above super-additive? Why?
2. Apply the **Shehory and Kraus algorithm** to find the best coalition structure. Assume that, everything else being equal, agents and coalitions are selected according to the lexicographic order.
3. Calculate the **Shapley values** for the agents in the largest coalition $\overline{C}$ found in the previous point.
4. Is the outcome formed by the Shapley values in the **core** for $\overline{C}$? Why?

1. No. For example, v({1,2,3,4}) = 0 < 8 = 4 + 4 = v({1,2}) + v({3,4}).

2. All agents run the same algorithm. Let us assume that they are synchronized.
   (a) Agents *i* (where *i* is in {1,2,3}) calculate C_i^*={1,2,3} and broadcast (*i*,{1,2,3})
   Agent 4 calculates C_4^*={2,3,4} and broadcasts (*4*,{1,2,4})
   (b) All agents form the set C^*={(*1*,{1,2,3}), (*2*,{1,2,3}), (*3*,{1,2,3}), (*4*,{1,2,4})} and calculate C_{max}={1,2,3}
   (c) All agents in {1,2,3} join C_{max}={1,2,3} and stop
   (d) Agent 4 forms coalition {4}

Decentralized Partially Observable MDPs

Definitions

- set of agents $I = 1, \dots, n$
- set of states S, *not known by the agents directly*
- set of actions A_i for each agent. $\bar{A}$ is the set of all joint actions
- $P : S \times S \times \bar{A} \to [0, 1]$
- set of observations for each agent i: Ω_i
- observation function $O : \bar{A} \times S \to \Delta\bar{\Omega}$
  - $O(\bar{o}|s, \bar{a})$: probability of receiving a joint observation when the world is in s and the joint action is $\bar{a}$
- reward function $R : S \times \bar{A}$

not so sure about this one:

$$V(s_0, \bar{q}) = R(s_0, \bar{a}) + \sum_{s', \bar{o}} P(s'|s_0, \bar{a}) O(\bar{o}|\bar{a}, s') V(s'|\bar{q}_{\bar{o}})$$

Model

Model

$I = \{1, 2\}$

$S = \{ \underbrace{TL}_{\text{tiger left}}, \underbrace{TR}_{\text{tiger right}} \}$

$\Omega_i = \{ \underbrace{hl}_{\text{hear tiger left}}, \underbrace{hr}_{\text{hear tiger right}} \}$

$A_i = \{ \underbrace{L}_{\text{listen}}, \underbrace{OL}_{\text{open left door}}, \underbrace{OR}_{\text{open right door}} \} \forall i \in I$

Goal: maximize the reward.

- -100 open door with tiger
- +100 open door with treasure
- -1 for listening

Transition function: prob of being in s and reach s' by doing $\vec{a}$:

s	$\vec{a}$	s'	$P(s'\|s, \vec{a})$
TL	$\langle$L, L$\rangle$	TL	1
TL	$\langle$L, L$\rangle$	TR	0
TL	$\langle$OR, L$\rangle$	TL	0,5
TL	$\langle$OR, L$\rangle$	TR	0,5
…	…	…	…

Observation function: prob of observing o being in s by doing $\vec{a}$:

$\vec{a}$	s	$\vec{o}$	$O(\vec{o}\|\vec{a}, s)$
$\langle$L, L$\rangle$	TL	$\langle$hl, hr$\rangle$	$\langle$0.85, 0.15$\rangle$
$\langle$L, L$\rangle$	TL	$\langle$hl, hl$\rangle$	$\langle$0.85, 0.85$\rangle$
…	…	…	…

Reward function: reward gained by doing $\vec{a}$ being in s:

$\vec{a}$	s	$R(\vec{a}, s)$
$\langle$L, L$\rangle$	TL	-2
$\langle$L, OR$\rangle$	TR	-101
$\langle$L, OR$\rangle$	TL	99
…	…	…

Constraint Optimization

DPOP - Dynamic Programming Algorithm

utils propagation

start from the bottom of the tree and report some "util messages" until you reach the root.

$$U_{i\to j}(Sep_i) = \max_{x_i}\left\{\left[(\otimes_{p\in Sep_i} F_{p,i})\right] \otimes U_{s\to i}(Sep_s)\right\}$$

where s is i's son in the tree, if it exists, otherwise don't consider U.

value propagation

start from the top and send down some "value messages".

the agent considered computes its best value as $x_i^* = \underset{x_i}{\operatorname{argmax}}\left\{U_{s\to i}(x_i) \otimes F_{f,i}(x_f^*, x_i)\right\}$.

where s always denotes i's son and f denotes i's father (if they don't exist don't consider them. the agent send to the son the message $V_{i\to s} = \{x_i^* = x_i^* \text{ value}\}$. in the set there are even other nodes' choices.

Side Notes on DPOP

- each agent send a util message to its only parent
- each agent sends a value message to all of its children
- induced width: the maximum number of parents and pseudo parents a node can have in a pseudo tree
- DPOP is always guaranteed to find the optimal solution

MaxSum Algorithm

Algorithm

1. Build the tree
2. at each iteration, each agent sends the following message to its neighbors

$$m_{i\to j}(x_j) = \alpha_{ij} + max_{x_i}\left(F_{ij(x_i,x_j)} + \sum_{A_k\in N_i - \{j\}} m_{k\to i}(x_i)\right)$$

   At the beginning, the terms in the sum are all set to zero.

3. The algorithm terminates when all the agents receive the same message twice.
4. When convergence is reached, each agent computes a local function z:

$$z_i(x_i) = \sum_{a_k\in N_i} m_{k\to i}(x_i)$$

5. agent i selects the value $\tilde{x}_i$ such that:

$$\tilde{x}_i = \underset{x_i}{\operatorname{argmax}}\left\{z_i(x_i)\right\}$$

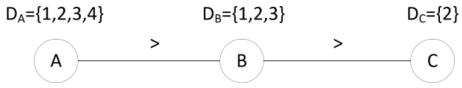
Side Notes on MaxSum

- I repeat: at a given step, each agent sends a message to each of its neighbors
- the size of a message sent by an agent a_i to its neighbor a_n depends only on the size of the domain of values from which a_n can pick
- suboptimal algorithm $\rightarrow$ in general no guaranteed to converge. if it does converge, it could do so in a local maximum.
- if the initial graph is acyclic, MaxSum converges to the optimal solution
- one could cut some arcs in order to make the graph acyclic and obtain an approximated solution

Asynchronous Backtracking

(he has never taught it I think).

Question 1 (8 points). Consider the following Distributed Constraint Satisfaction Problem (DCSP), in which there are three variables/agents (A, B, and C), whose domains are reported below, and two constraints (A > B and B > C).



1) Consider the priority order A – B – C and solve the problem using the **asynchronous backtracking** algorithm. Assume that the agents activates in cyclic turns A – B – C – A – … and that, when an agent activates, it receives *all* the pending messages in the order in which they have been sent. Everything else being equal, agents select their values in increasing order.
2) Consider now the priority order C – B – A and solve the same problem, with the same algorithm and the same assumptions as above.

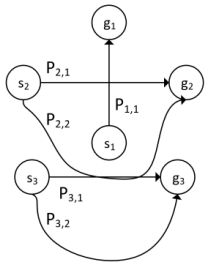
1) A possible computation is as follows.

- A assigns 1 and sends ok?(A,1) to B
- B receives ok?(A,1) from A, cannot assign any legal value, sends nogood({A,1}) to A, assigns 1, and sends ok?(B,1) to C
- C receives ok?(B,1) from B, cannot assign any legal value, sends nogood({B,1}) to B, and assigns 2
- A receives nogood({A,1}) from B, assigns 2, and sends ok?(A,2) to B
- B receives nogood({B,1}) from C, assigns 2, sends ok?(B,2) to C, receives ok?(A,2) and from A, cannot assign any legal value, sends nogood({A,2}) to A, keeps 2
- C receives ok?(B,2) from B, cannot assign any legal value, sends nogood({B,2}) to B, and keeps 2
- A receives nogood({A,2}) from B, assigns 3, and sends ok?(A,3) to B
- B receives nogood({B,2}) from C, assigns 3, sends ok?(B,3) to C, receives ok?(A,3) and from A, cannot assign any legal value, sends nogood({A,3}) to A, and keeps 3
- C receives ok?(B,3) from B and keeps 2
- A receives nogood({A,3}) from B, assigns 4, and sends ok?(A,4) to B
- B receives ok?(A,4) and from A and keeps 3

2) A possible computation is as follows.

- C assigns 2 and sends ok?(C,2) to B
- B receives ok?(C,2) from C, assigns 3, and sends ok?(B,3) to A
- A receives ok?(B,3) from B and assigns 4

Question 1 (7 points). Consider the following situation in which three robots (1, 2, and 3) can move from their starting locations (s_i) to their goal locations (g_i) only along one of their possible paths ($P_{i,j}$):



Each robot i must choose one of its paths $P_{i,j}$ in such a way that the paths chosen by any two robots do not intersect.

1. Formulate the above situation as a **Distributed Constraint Satisfaction Problem** (**DCSP**), specifying agents (variables), domains, and constraints.
2. Solve the DCSP formulated above using the **asynchronous backtracking algorithm**. Assume that the priority order of the agents corresponds to their lexicographic order and that, everything else being equal, values are selected according to the lexicographic order. Show the trace of the execution of the algorithm when agents activate sequentially in turns, starting from the highest priority agent.
3. What is the content of the local_views of the agents at the end of the execution of the algorithm?

1. Agents (variables): the three robots R_1, R_2, and R_3.
   Domains: $D_1=\{P_{1,1}\}$, $D_2=\{P_{2,1},P_{2,2}\}$, and $D_3=\{P_{3,1},P_{3,2}\}$.
   Constraints (forbidden combinations): $C_{12}=\{(P_{1,1},P_{2,1})\}$ and $C_{23}=\{(P_{2,2},P_{3,1})\}$.

2. Agent R_1: assigns $R_1=P_{1,1}$ and sends ok?(R_1,$P_{1,1}$) to R_2
   Agent R_2: receives ok?(R_1,$P_{1,1}$), assigns $R_2=P_{2,2}$, and sends ok?(R_2,$P_{2,2}$) to R_3
   Agent R_3: receives ok?(R_2,$P_{2,2}$), assigns $R_3=P_{3,2}$

3. local_view of R_1 is empty, local_view of R_2 is {(R_1,$P_{1,1}$)}, and local_view of R_3 is (R_2,$P_{2,2}$)}.

Combinatorial Auctions

Branch and Bound

Question 4 (8 points). Consider a combinatorial auction with five items (A, B, C, D, and E) in which the auctioneer has received the following bids:

items	bid
{C}	3
{D}	4
{B,D}	5
{B,E}	7
{A,B,C}	5
{A,C,D}	7
{A,B,C,E}	9

For each issue, calculate the heuristic value to be employed in the **branch-and-bound algorithm**. Apply such algorithm, reporting the search tree, the places where the algorithm cuts parts of the tree, and the optimal set of bids returned by the algorithm. Everything else being equal, consider bids in the lexicographic order.

The heuristic values for the items are:

$$h_A = \max\left\{\frac{5}{3}, \frac{7}{3}, \frac{9}{4}\right\} = \frac{7}{3}$$

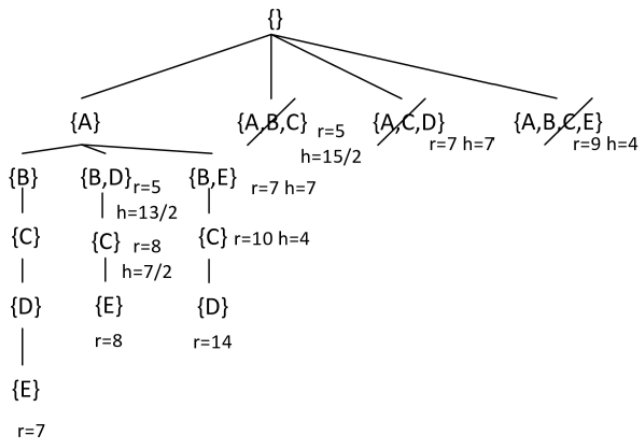
$$h_B = \max\left\{\frac{5}{2}, \frac{7}{2}, \frac{5}{3}, \frac{9}{4}\right\} = \frac{7}{2}$$

$$h_C = \max\left\{3, \frac{5}{3}, \frac{7}{3}, \frac{9}{4}\right\} = 3$$

$$h_D = \max\left\{4, \frac{5}{2}, \frac{7}{3}\right\} = 4$$

$$h_E = \max\left\{\frac{7}{2}, \frac{9}{4}\right\} = \frac{7}{2}$$

The search tree build by branch and bound is:



The optimal set of bids is thus: {A}, {B,E}, {C}, and {D}.

Multiagent Learning

General

- now we have a reward function for each agent
- why *learning*? because agents don't know in advance their reward function and transition function
- Factoredness
  the local goal of an agent is aligned to the global goal $\rightarrow$ actions that optimize a reward will optimize other rewards as well
- Learnability
  how discernible the impact of an action is on an agent's reward function $\rightarrow$ higher learnability means it is easier for an agent to take actions that maximize its reward
- factoredness and learnability are complementary concepts
- Multiagent Reinforcement Learning
  - is modeled with Markov Stochastic Games
  - accepts always at least one deterministic Nash Equilibrium if actions are deterministic (we'll see only the case in which actions are deterministic)
  - can be seen as
    - MDP with multiple agents
    - POMDP with no partial observations

Stochastic Markov Game

defined by:

- a set of states S
- a set of actions for each agent i A^i . Consequently, the set of join actions $\bar{A}$ is defined as $\bar{A} = A^1 \times \cdots \times A^n$
- a transition function $p : S \times A^1 \times \cdots \times A^n \rightarrow \Delta(S)$.
- a reward function r^i for each agent i, defined as $r^i : S \times A^1 \times A^2 \times \cdots \times A^n \rightarrow \mathbb{R}$
- a policy π^i for each agent i defined as $\pi^i : S \rightarrow A^i$ (policies will be considered only as deterministic for us)
- an optimal policy π^i_* for each agent i that maximizes its reward
- value functions v^i, defined below

now we assume the agents don't know the r^i and p!

Value Function

$$v^i(s, \pi^1, \pi^2, \ldots, \pi^n) = \sum_t \beta^t E[r^i_t | \pi^1, \pi^2, \ldots, \pi^n, s_0 = s]$$

Our goal is to maximize v^i for all agents.

- big plus!
  When the environment is deterministic (the transition function is deterministic) it is really easy to compute function v^i because we get rid of the expected value (we have an exact value for each timestep).

Nash Equilibrium

a set of joint policies$(\pi_*^1, \dots, \pi_*^n)$ such that

$$v^i(s, \pi_*^1, \dots, \pi_*^i, \dots \pi_*^n) \geq v^i(s, \pi_*^1, \dots, \pi^i, \dots \pi_*^n)$$
$$\forall \pi^i, \forall s, \forall i$$

- The goal is to learn policies that are in equilibrium.
- Reach an equilibrium state is a goal, there is not global utility.
- It is possible to prove that each stochastic game admits a NE.

Nash-Q Function

Q-function

the Q function of an agent i is a function learnt by all the agents that returns a number for any combination of states s and joint actions $\bar{a}$.

Nash Q-function

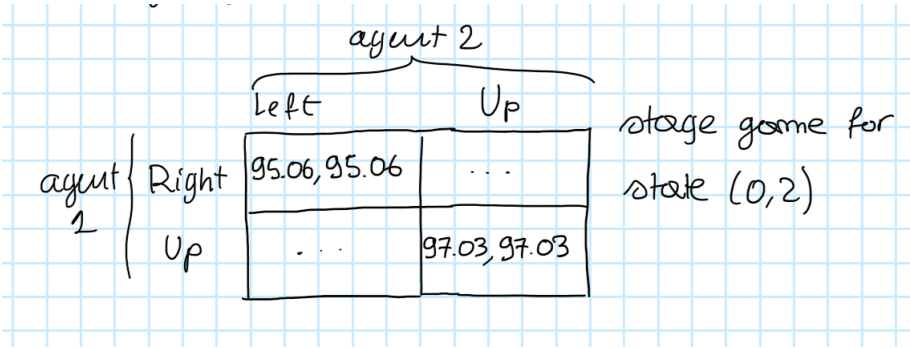
Similar to the Q-function, but it refers to equilibrium policies $\pi_*^1, \dots, \pi_*^n$

$$Q_*^i(s, a^1, \dots, a^n) = r^i(s, a^1, \dots, a^n) + \beta \sum_{s' \in S} P(s'|s, a^1, \dots, a^n) v^i(s', \pi_*^1, \dots, \pi_*^n)$$

actions performed $a^1, \dots, a^n$ can be any actions, no need to be sampled by the respective π_*.

- r_i is the reward function of agent i
- β is the discount factor
- p is the transition function
- v^i is the value function of agent i (i.e. what agent i tries to maximize)

Important:
if you calculate the Nash Q-function in a given state s, you are implicitly defining a strategic game in normal form that is a Stage Game:



Nash-Q Learning Algorithm

Following the Nash-Q Learning algorithm , the Q-functions of the agents converge to the corresponding Nash-Q functions.

- each agent, in order to locally update the Q-values during the execution of the algorithm, needs to know
  - the state s
  - the actions performed by all the agents
  - the reward received by all the agents
- *learning* because: rewards and transition functions are not known in advance, so agents need to figure it out
- every agent keeps the Q-Nash value of other agents updated
- build a stage game table for every state

	Left	Up
Right	0,0	0,0
Up	0,0	0,0

State (0,2)

	Up	Left	Down
Up	0,0	0,0	0,0
Right	0,0	0,0	0,0
Down	0,0	0,0	0,0

State (3,5)

Algorithm

1. initialization phase: $Q^i_{t=0}(s, a^1, \dots, a^n) = 0 \quad \forall s \in S, \forall a^i \in A^i \ \ \forall i$
2. each agent will perceive the state of the world and decide what to do
3. each agent receives its reward and other agents' rewards with the actions they performed
4. update Q-function

$$\text{known to both agents} \begin{bmatrix} a^1_{t=0} = \text{Right} \\ a^2_{t=0} = \text{Left} \\ r^1_{t=0} = -1 \\ r^2_{t=0} = -1 \end{bmatrix}$$

Initialization

set all to 0:

$Q^1_0(s, a^1, a^2) = 0$

$Q^2_0(s, a^1, a^2) = 0$

Formula

The Q-function of an agent is updated as:

$$Q^i_{t+1}(s,a^1,\dots,a^n) = (1-\alpha_t)Q^i_t(s,a^1,\dots,a^n) + \alpha_t\left(r^i_t + \beta\,\text{Nash}Q^i_t(s')\right)$$

Observed Reward

The payoff that agent i gets for the NE of the stage game that is defined by the current Q-function

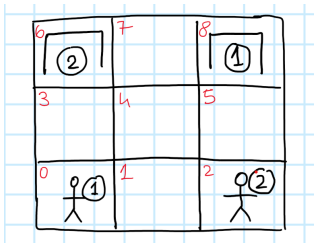
It is possible to prove that this converges to Q-functions that are the optimal Q-functions Q^* (corresponding to the NE).

In general a game can have multiple $NE \rightarrow$ the same equilibrium is chosen by all the agents.

α_t can have multiple forms, a popular one is the following:

$$\alpha_t = \frac{1}{x(s,a^1,\dots,a^n)}$$

Example



the initial situation, shown only for two of the possible states:

	Left	Up
Right	0,0	0,0
Up	0,0	0,0

State (0,2)

	Up	Left	Down
Up	0,0	0,0	0,0
Right	0,0	0,0	0,0
Down	0,0	0,0	0,0

State (3,5)

now suppose:

$s_0 = (0,2)$

$\bar{a} = \{\text{Right},\text{Left}\}$

$\beta = 0.99$

$r^1_t = r^2_t = -1$

$$Q^1_{t+1}(s_0,\bar{a}) = (1-\alpha_t)\cdot 0 + \alpha_t(-1+0.99\cdot 0) = -\alpha_t$$

where 0 is the utility of agent 1 for the chosen NE (we have chosen randomly one of the 4 NE, let's say the one on the top left).

$$Q^2_{t+1}(s_0,\bar{a}) = \dots = -\alpha_t$$

	Left	Up
Right	$-\alpha_t$, $-\alpha_t$	0,0
Up	0,0	0,0

Question 4 (8 points). Consider the Nash-Q learning algorithm for a stochastic game.
(a) Describe the difference between the Q-functions that are learnt by the agents and the corresponding Nash Q-functions. Describe the meaning of all symbols appearing in your answer.
(b) What does an agent need to know in order to locally update Q-values during the execution of the algorithm? Why?
(c) Consider a stochastic game with $|S| = 3$ states and $n = 3$ agents, with $|A^1| = 3$ actions, $|A^2| = 2$ actions, and $|A^3| = 4$ actions, respectively. How many Q-values does each agent need to store during the execution of the algorithm? Why?

(a) The Q-functions are learnt by the agents and return a number (value) for any combination of states s and joint actions $a^1, a^2, \dots, a^n$:

$$Q_t^i(s, a^1, a^2, \dots, a^n)$$

The Nash Q-functions:

$$Q_*^i(s, a^1, a^2, \dots, a^n) = r^i(s, a^1, a^2, \dots, a^n) + \beta \sum_{s' \in S} p(s'|a^1, a^2, \dots, a^n) v^i(s', \pi_*^1, \pi_*^2, \dots, \pi_*^n)$$

have a similar signature but refer to equilibrium policies $\pi_*^1, \pi_*^2, \dots, \pi_*^n$. In the above formula, r^i is the reward function of agent i, β is the discount factor, $p()$ is the transition function, and v^i is the value function of agent i (i.e., what agent i tries to maximize).
Following the Nash-Q algorithm, the Q-functions of the agents converges to the corresponding Nash-Q functions.

(b) Each agent needs to know the state, the actions performed by all the agents, and the rewards received by all the agents.
Indeed, the update rule is:

$$Q_{t+1}^i(s, a^1, a^2, \dots, a^n) = (1 - \alpha_t) Q_t^i(s, a^1, a^2, \dots, a^n) + \alpha_t [r_t^i + \beta \text{Nash}(s')]$$

where r_t^i is the reward received by agent i at time t and $\text{Nash}(s') = Q_t^i(s', \pi^1(s'), \pi^2(s'), \dots, \pi^n(s'))$ is the payoff of agent i in the Nash equilibrium of the stage game in s'.
In addition, agents need to know and follow the same rule to break ties and selecting the same Nash equilibria in the stage games.

(c) The Q-function for each agent i is $Q_t^i(s, a^1, a^2, a^3)$ and has $3 \times 3 \times 2 \times 4 = 72$ entries. Each agent has to store the Q-functions of all the agents, thus in total $72 \times 3 = 216$.

Never Done

Common Knowledge

Example 1

Question 2 (7 points). Consider a **common knowledge** situation in which two agents, A and B, are operating in a world whose state is decided by throwing a die. The possible states of the world are thus: $S = \{s_1,s_2,s_3,s_4,s_5,s_6\}$.
(1) Assume that, for agent A, states $\{s_1,s_2\}$ are indistinguishable from each other. Similarly for states $\{s_3,s_4,s_5\}$. Report the *information (perception) function* of agent A.
(2) Consider the event E_1 = "the state of the world is not larger than 3". In which states $K_A(E_1)$ does agent A know event E_1?
(3) Assume that agent B cannot distinguish between states $\{s_2,s_3\}$ and $\{s_4,s_5\}$. Consider the event E_2 = "agent A knows E_1". In which states $K_B(E_2)$ does agent B know E_2?
(4) Consider now event E_3 = "agent B knows E_2" = "agent B knows that agent A knows E_1". Is event E_3 *self-evident* for agents A and B?
(5) Is event E_3 *common knowledge* between agents A and B in state s_1? Explain why.

(1) The perception function of agent A is:
$P_A(s_1) = \{s_1,s_2\}$
$P_A(s_2) = \{s_1,s_2\}$
$P_A(s_3) = \{s_3,s_4,s_5\}$
$P_A(s_4) = \{s_3,s_4,s_5\}$
$P_A(s_5) = \{s_3,s_4,s_5\}$
$P_A(s_6) = \{s_6\}$

$K_B(E_2) = \{s_1\}$
$K_B[E_3] = K_B[K_A(E_2)]$

(2) $K_A(E_1) = K_A(\{s_1,s_2,s_3\}) = \{s_1,s_2\}$

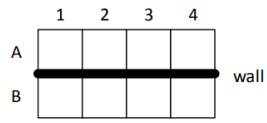
(3) The partition $\wp_B$ induced on the set of states S by the perception function of agent B is:
$\wp_B = \{ \{s_1\}, \{s_2,s_3\}, \{s_4,s_5\}, \{s_6\} \}$
And $K_B(E_2) = K_B(K_A(E_1)) = K_B(\{s_1,s_2\}) = \{s_1\}$.

(4) $K_A(E_3) = K_A(\{s_1\}) = \{\}$, not self-evident.
$K_B(E_3) = K_B(\{s_1\}) = \{s_1\} = E_3$, self-evident.

(5) Being E_3 not self-evident for agent A and being composed of a single state, it cannot be common knowledge between agents A and B.

Example 2

Question 2 (8 points). Consider the following environment:



It is known that an object is placed above the wall and another object is placed below the wall with the following constraint: if the object is placed at p ($p\in\{1,2,3,4\}$) above the wall, the other one can be only placed at p-1, p, or p+1 below the wall (if they are all valid places of the environment), and vice versa. Agent A can only see above the wall and agent B can only see below the wall. The situation is modeled as an **interactive thinking** setting.

(1) Define the set S of possible states of the world.
(2) Define the perception functions P_A and P_B of the two agents.
(3) Define the event E="at least one of the two objects is at place 2".
(4) Assuming that the actual situation s is that the object above the wall is at place 2 and that below the wall is at place 3, does agent A know event E in state s?
(5) Is event E self-evident for agent A?
(6) Is event E common knowledge between the two agents?

(1) A state is xy, with the meaning that x is the place of the object above the wall and y is the place of the object below the wall. The set S of possible states of the world is:
S={11,12,21,22,23,32,33,34,43,44}.

(2) $P_A(xy)$={all states xz such that $z\in\{x-1,x,x+1\}$} and
$P_B(xy)$={all states zy such that $z\in\{y-1,y,y+1\}$}.

(3) E={12,21,22,23,32}

(4) s=23 and agent A knows E in s because $P_A(23)=\{21,22,23\}\subseteq\{12,21,22,23,32\}=E$.

(5) $K_A(E)=\{21,22,23\}\neq E$ and so E is not self-evident for agent A.

(6) No, because in s=23, $P_B(23)=\{23,33,43\}\not\subset E$ and so B does not know E in s.