Multiagent decision-making and control

Dynamic games

Maryam Kamgarpour

Professor of Engineering (IGM, STI), EPFL

ME-429, Spring 2025, EPFL

Course topics

1. Static games
2. Zero-sum games
3. Potential games
4. Extensive form games
5. Dynamic games, dynamic programming principle
6. Dynamic games, dynamic programming for games
7. Dynamic games, linear quadratic games, Markov games
8. Convex games, Nash equilibria characterization
9. Convex games, Nash equilibria computation
10. Auctions
11. Bayesian games
12. Learning in games
13. Final project presentations

Admin matter

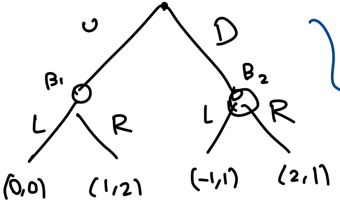
- Quiz 1 feedback/discussion
- Timeline: project update: 13:15-14:00, 10% , April 16

A brief peak into leader-follower games

= Stackelberg game

- Sequential game in one-stage with full information
- examples: government setting a policy, society following, security game, ...

players are minimizer



matrix form ⇒

	LL	LR	RL	RR
U	(0,0)	(0,0)	(1,2)	(1,2)
D	(-1,1)	(2,1)	(-1,1)	(2,1)

Nash equilib

⇓ backward induction

B1 : P2 will choose L

B2 : P2 can choose L,R (indifferent)

if P1 thinks P2 chooses L in B2 ⇒ P1 chooses D : (-1,1)

else, P1 chooses, U : (0,0)

- Recall Quiz 1, Problem 2, part (c): in the zero-sum setting, the second player was better off
- What about general (non zero-sum case?)

Review: last lecture

mixed

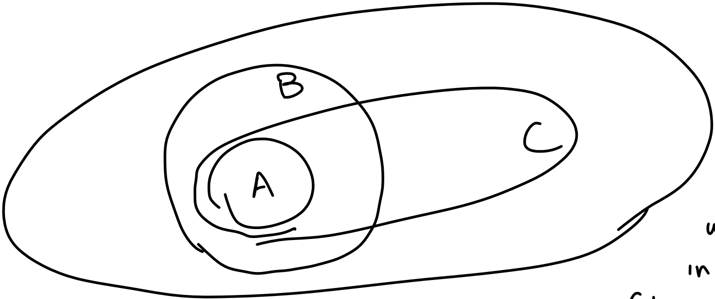
- ~~Randomized~~ strategies in extensive form games
- Behavioral strategies in feedback games

D : set of all mixed NE in extensive form games

C : set of all behavioral strateg NE (feedback games)

B : " " " pure strategy NE in extensive form..

A : " " " subgame perfect NE (& well-defined in feedback games)



$A \subset B$: we proved in Lecture 5 (Lemma 7.1 in Hespanha)

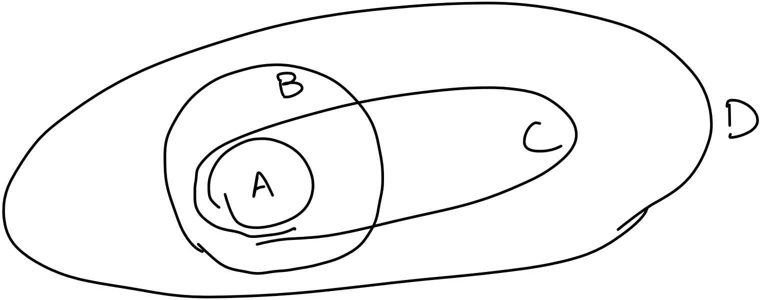
Review: last lecture

$\checkmark$ $A \subset B$

$\checkmark$ $B \subset D$

,, D

- Randomized strategies in extensive form games
- Behavioral strategies in feedback games : C
- subgame perfect equil. : A



$A \subset C$ follows from the fact that every subgame pure strategy $\subset$ mixed strategy

$\underbrace{C \subset D}$ we haven't proven. it requires a result called Kuhn's theorem.

Review: last lecture

- Randomized strategies in extensive form games
- Behavioral strategies in feedback games

D : set of all mixed NE in extensive form games

C : set of all behavioral strateg NE (feedback games)

B : " " " pure strategy NE in extensive form..

A : " " " subgame perfect NE (well-defined in feedback games)

* A can be empty even when B is non-empty

  ex : Example 7.4 in Hespanha's book & last lecture

* C is always nonempty (follows from Lecture 7) for every subgame

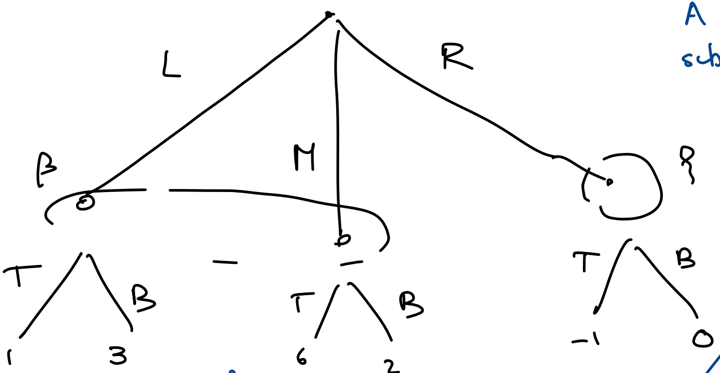
* $C \subset D$ $\subset$: is strict,

  see Example 7.4 in Hespanha.

* $\exists$ NE in C and not in B.

Relationship between equilibria in extensive form games

zero-sum game, P1 minimizer



A strict subset of B.

subgame on Left

	T	B
L	1	3
M	6	2

no pure strategy subgame perfect NE

but it has pure strategy NE

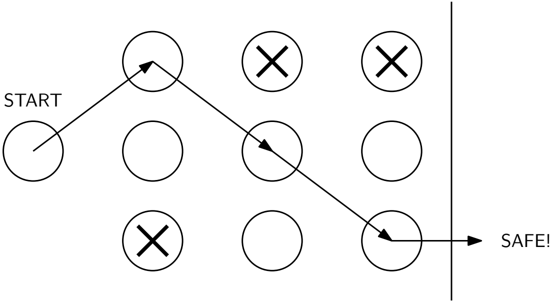
	TT	TB	BT	BB
L	1	1	3	3
M	6	6	2	2
R	-1	0	-1	0

NE

This lecture

Dynamic programming

Motivation: escape game



Player 1 (Alice) is trying to escape, going from the **start node** to the **safe zone** without being intercepted. At every stage of the game, Alice moves 1 step closer to the safe zone. She can decide to continue on the same row, or instead move diagonally one row up or one row down.

Player 2 (Eve) is trying to stop Alice. At each stage, Eve is aware of Alice's current position, and she is allowed to block one of the three rows. If she selects the row corresponding to Alice's next move, she stops her and she wins the game.

Escape game

1 stage

Consider the case with 3 rows and only 1 stage (that is, only 1 chance for Eve to stop Alice). What is the corresponding tree? What is the Nash equilibrium strategy for Alice and for Eve? What is the value of the game?

Escape game

1 stage

Consider the case with 3 rows and only 1 stage (that is, only 1 chance for Eve to stop Alice). What is the corresponding tree? What is the Nash equilibrium strategy for Alice and for Eve? What is the value of the game?

2 stages

Consider the case where there are 3 rows and 2 stages (that is, 2 chances for Eve to stop Alice). How many LP do we need to solve?

Escape game

1 stage

Consider the case with 3 rows and only 1 stage (that is, only 1 chance for Eve to stop Alice). What is the corresponding tree? What is the Nash equilibrium strategy for Alice and for Eve? What is the value of the game?

2 stages

Consider the case where there are 3 rows and 2 stages (that is, 2 chances for Eve to stop Alice). How many LP do we need to solve?

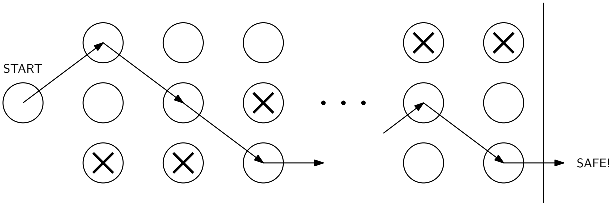
3 stages

Consider the case where there are 3 rows and 3 stages (like in the previous figure). How many LP do we need to solve?

Escape game

Multi-stage game with

- finite actions
- large number of stages (20)

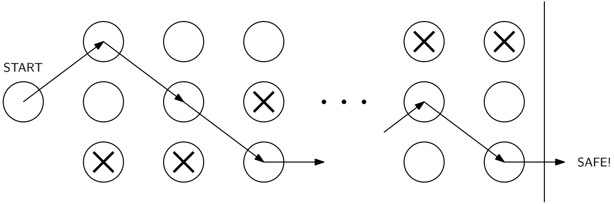


How many “Player 1 information sets” in the tree that represents this game?

Escape game

Multi-stage game with

- finite actions
- large number of stages (20)



How many “Player 1 information sets” in the tree that represents this game?

Order of: $1 + 9 + 81 + \ldots + 3^{20}$ linear programs

Taming the complexity of dynamic games

How can you solve this game for 20 stages?

Taming the complexity of dynamic games

How can you solve this game for 20 stages?

- **Key idea:** In every “position”, the probability of escaping (and the optimal strategy) does not depend on the past decisions.
- We can perform backward induction on the “positions” (**states!**) rather than on the game tree.

Taming the complexity of dynamic games

How can you solve this game for 20 stages?

- **Key idea:** In every “position”, the probability of escaping (and the optimal strategy) does not depend on the past decisions.
- We can perform backward induction on the “positions” (**states!**) rather than on the game tree.

$$3 \times 20 \text{ linear programs instead of } 5 \times 10^9$$

With more actions available at each stage, the speed up is even larger!

Taming the complexity of dynamic games

How can you solve this game for 20 stages?

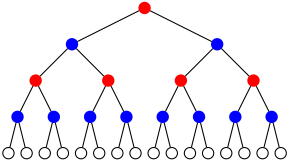
- **Key idea:** In every “position”, the probability of escaping (and the optimal strategy) does not depend on the past decisions.
- We can perform backward induction on the “positions” (**states!**) rather than on the game tree.

$$3 \times 20 \text{ linear programs instead of } 5 \times 10^9$$

With more actions available at each stage, the speed up is even larger!

We will adapt this idea to dynamic games with **infinite** action spaces.

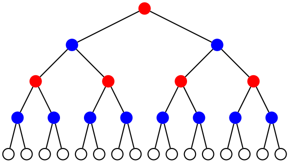
From tree model from loop model



Consider a **feedback game** in extensive form in which at each stage k

- Player 1 action: $u_k \in \mathcal{U}_k \subseteq \mathbb{R}^m$
- Player 2 action: $v_k \in \mathcal{V}_k \subseteq \mathbb{R}^n$

From tree model from loop model



Consider a **feedback game** in extensive form in which at each stage k

- Player 1 action: $u_k \in \mathcal{U}_k \subseteq \mathbb{R}^m$
- Player 2 action: $v_k \in \mathcal{V}_k \subseteq \mathbb{R}^n$

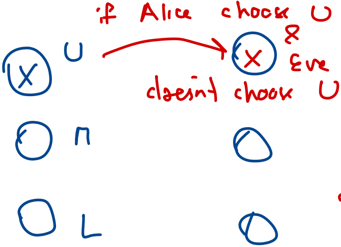
Assumption 1

There exists a **state** that evolves at each stage

$$x_{k+1} = f(x_k, u_k, v_k)$$

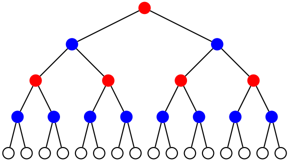
We can "attach" a state x_k to each Player 1 node.

Alice is here at stage k



x_{k+1} position of Alice in next stage is a function f of what Alice & Eve choose at stage k

From tree model from loop model



Consider a **feedback game** in extensive form in which at each stage k

- Player 1 action: $u_k \in \mathcal{U}_k \subseteq \mathbb{R}^m$
- Player 2 action: $v_k \in \mathcal{V}_k \subseteq \mathbb{R}^n$

Assumption 1

There exists a **state** that evolves at each stage

$$x_{k+1} = f(x_k, u_k, v_k)$$

We can "attach" a state x_k to each Player 1 node.

Assumption 2

$$\text{outcome: } \sum_{k=1}^{K} g_k(x_k, u_k, v_k)$$

From tree model to loop model

characterize

the game

$$x_{k+1} = f(x_k, u_k, v_k), \qquad \text{outcome: } \sum_{k=1}^{K} g_k(x_k, u_k, v_k)$$

The value of a subgame rooted in a Player 1 node at stage k' is then

$$\sum_{k=k'}^{K} g_k(x_k, u_k, v_k)$$

and therefore depends only on the value x_k attached to that node.

if zero sum

P1 aims to minimize $\sum_{k=1}^{K} g_k(x_k, u_k, v_k)$

& P2 ~~a~~ " maximize "

subject to the dynamics $x_{k+1} = f(x_k, u_k, v_k)$

From tree model to loop model

$$x_{k+1} = f(x_k, u_k, v_k), \qquad \text{outcome: } \sum_{k=1}^{K} g_k(x_k, u_k, v_k)$$

The value of a subgame rooted in a Player 1 node at stage k' is then

$$\sum_{k=k'}^{K} g_k(x_k, u_k, v_k)$$

and therefore depends only on the value x_k attached to that node.

At each stage, the state is a sufficient representation of the game for all purposes.

Information structure

$$x_{k+1} = f(x_k, u_k, v_k), \qquad \text{outcome: } J(x_1, u_1, \ldots, u_K, v_1, \ldots, v_K) = \sum_{k=1}^{K} g_k(x_k, u_k, v_k)$$

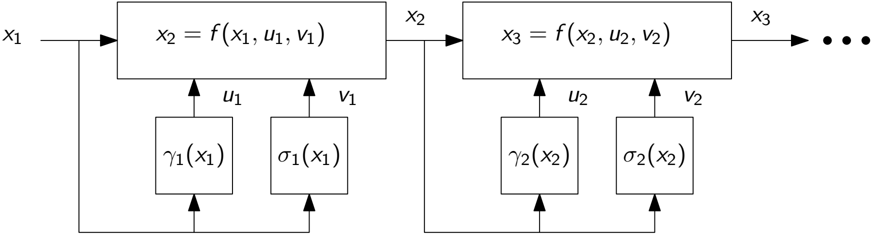
There are different assumptions one can make regarding the information each player uses to compute its action. We will focus on two options:

- Open-loop: $u_k(x_1), v_k(x_1)$, for $k = 1, 2, \ldots, K$
  Note: for a given sequence of actions of each player, the cost depends on the initial state x_1.
- Perfect state-feedback $u_k(x_{\cancel{1}K}), v_k(x_{\cancel{1}K})$, for $k = 1, 2, \ldots, K$

Note: “perfect” highlights players have exact knowledge of x_k at each stage. In practice, players might have same noisy measurements of the state x_k (partial observation) or each player has its own noisy measurement (partial and asymmetric observation).

Loop model

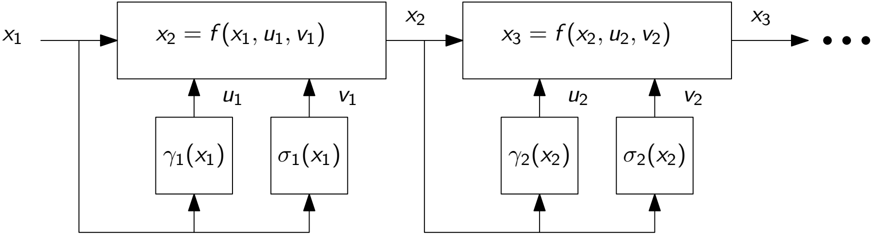
For the remainder we consider perfect state-feedback information structure.



- **Stages** become **discrete time**

Loop model

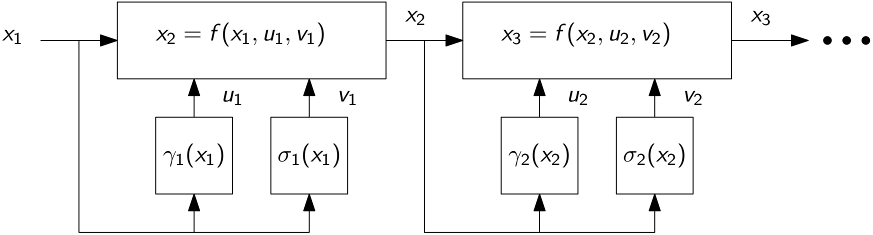
For the remainder we consider perfect state-feedback information structure.



- **Stages** become **discrete time**
- **Strategies** (maps from information sets to actions) become **Feedback laws** (maps from states to inputs)

Loop model

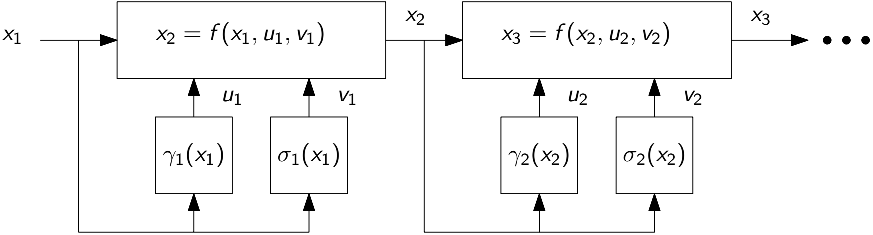
For the remainder we consider perfect state-feedback information structure.



- **Stages** become **discrete time**
- **Strategies** (maps from information sets to actions) become **Feedback laws** (maps from states to inputs)
- Players have the same information available (**simultaneous play**)

Loop model

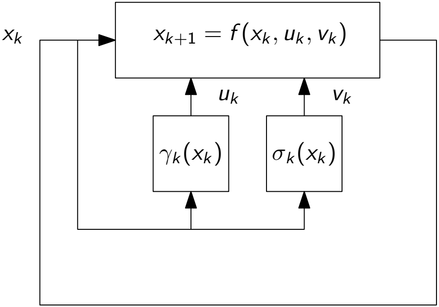
For the remainder we consider perfect state-feedback information structure.



- **Stages** become **discrete time**
- **Strategies** (maps from information sets to actions) become **Feedback laws** (maps from states to inputs)
- Players have the same information available (**simultaneous play**)
- Variations are possible (output feedback, delays, ...)

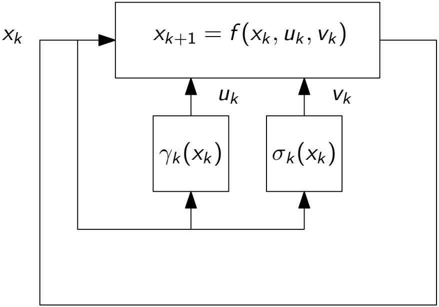
Loop model

Just for convenience of representation, we loop it around to get



Loop model

Just for convenience of representation, we loop it around to get

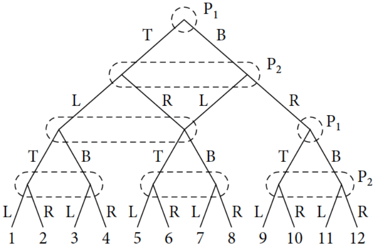


Basic elements

- Stages $k = 1, \ldots, K$
- State space $x_k \in \mathcal{X}$ and state evolution map $f(x_k, u_k, v_k)$
- Action space $u_k \in \mathcal{U}$, $v_k \in \mathcal{V}$
- Outcome function $g_k(x_k, u_k, v_k)$

Loop model generality

The tree model can be converted to loop model.
The loop model can also define non-tree extensive form games. See for examples below (Hespanha Figure 14.1b):



Note: in the example above a player may have different costs depending on how the end of the game is reached

Feedback game or feedback control?

What are the information sets?

- We assumed **feedback games**, so x_k is known to both Players at stage k
  - we attached the state to Player 1 nodes, which are roots of separate subtrees
- We also assume **simultaneous play**.

Feedback game or feedback control?

What are the information sets?

- We assumed **feedback games**, so x_k is known to both Players at stage k
  - we attached the state to Player 1 nodes, which are roots of separate subtrees
- We also assume **simultaneous play**.

Is it **necessary** to have a feedback game?
What are the information sets implied by the loop model?

Non-zero sum dynamic games

A three-truck platoon

The state $x_k \in \mathbb{R}^2$ describes the 2 distances $d_k^{(1)}, d_k^{(2)}$ between trucks in the platoon and their speeds $s_k^{(1)}, s_k^{(2)}$.
D is the desired distance.
Inputs u_k and v_k are accelerations.

Followers in the platoon want to minimize their own cost functionals

$$g^{(1)}(x_k, u_k, v_k) = (d_k^{(1)} - D)^2 + (d_k^{(2)} - D)^2 + u_k^2$$

$$g^{(2)}(x_k, u_k, v_k) = (d_k^{(2)} - D)^2 + v_k^2$$

State evolution:

$$d_{k+1}^{(1)} = d_k^{(1)} - T(s_k^{(2)} - s_k^{(1)}), \quad s_{k+1}^{(1)} = s_k^{(1)} + Tu_k, \ldots$$

All results extend

The definitions (NE, pure strategies, mixed strategies) extend to the loop model, and have a “control” interpretation.
Both players know exactly the state x_k of the game at the entry of the current stage k and can use this information to compute their actions (also referred to as **perfect information** game)

Subgame-perfect Nash Equilibrium

A strategy is a **subgame perfect** equilibrium if it represents a NE of every subgame of the original game.

Subgame-perfect NE strategies (that is, feedback laws) are what in control, in the context of single player decision-making, we define as **optimal feedback laws**.

All results extend

The definitions (NE, pure strategies, mixed strategies) extend to the loop model, and have a "control" interpretation.
Both players know exactly the state x_k of the game at the entry of the current stage k and can use this information to compute their actions (also referred to as **perfect information** game)

Subgame-perfect Nash Equilibrium

A strategy is a **subgame perfect** equilibrium if it represents a NE of every subgame of the original game.

Subgame-perfect NE strategies (that is, feedback laws) are what in control, in the context of single player decision-making, we define as **optimal feedback laws**.

$$u_k = \gamma_k(x_k) \qquad v_k = \sigma_k(x_k)$$

Backward induction · · · lets consider 0 · sum game .

STEP K**:** Consider all the infinite subgames rooted in x_K:

(Notice the abuse of notation: P1 node / state)

$$x_{K+1} = f(x_K, u_K, v_K)$$

with outcome

$$g_K(x_K, u_K, v_K)$$

Determine γ_K^*, σ_K^* (functions of x_K) such that

$$g_K(x_K, \gamma_K^*(x_K), \sigma_K(x_K)) \;\leq\; g_K(x_K, \gamma_K^*(x_K), \sigma_K^*(x_K)) \;\leq\; g_K(x_K, \gamma_K(x_K), \sigma_K^*(x_K))$$

Backward induction

Value function $V_K(x_K)$: value of the subgame rooted in x_K, that is

$$V_K(x_K) = g_K(x_K, \gamma_K^*(x_K), \sigma_K^*(x_K))$$

STEP $K-1$**:** Consider all the infinite subgames rooted in x_{K-1}

$$x_K = f(x_{K-1}, u_{K-1}, v_{K-1})$$

with outcome

$$\sum_{k=K-1}^{K} g_k(x_k, u_k, v_k)$$

which we rewrite as

$$g_{K-1}(x_{K-1}, u_{K-1}, v_{K-1}) + V_K(x_K)$$

and therefore

$$g_{K-1}(x_{K-1}, u_{K-1}, v_{K-1}) + V_K(f(x_{K-1}, u_{K-1}, v_{K-1}))$$

Backward induction

$$g_{K-1}(x_{K-1}, u_{K-1}, v_{K-1}) + V_K(f(x_{K-1}, u_{K-1}, v_{K-1}))$$

Determine $\gamma^*_{K-1}, \sigma^*_{K-1}$ (functions of x_{K-1}) that are saddle-points for

$$g_{K-1}(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1})) + V_K(f(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1})))$$

and so on, backward until **stage 1**.

2-player zero-sum case: backward induction

Value function $V_K(x_K)$: value of the subgame rooted in x_K, that is

$$V_K(x_K) = g_K(x_K)$$

STEP $K-1$**:** Consider all the subgames rooted in x_{K-1}

$$x_K = f(x_{K-1}, u_{K-1}, v_{K-1})$$

with outcome

$$\sum_{s=K-1}^{K-1} g(x_s, u_s, v_s) + g_K(x_K)$$

Find (if they exist) $\gamma^*_{K-1}, \sigma^*_{K-1}$ (functions of x_{K-1}) as the saddle-points for

$$g(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1})) + V_K(f(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1})))$$

If at any iteration, the above does not have a saddle-point equilibrium, the procedure fails.

in pure strategy

- There may be other Nash equilibria for the game (consider the dynamic games given in Figure 7.4 of Hespanha)

2-player general-sum case: backward induction

Player i's value function $V_K^i(x_K)$: player i's value of the subgame rooted in x_K

$$V_K^i(x_K) = g_K^i(x_K),\ i = 1, 2.$$

STEP $K-1$**:** Consider the subgames rooted in x_{K-1}

$$x_K = f(x_{K-1}, u_{K-1}, v_{K-1})$$

with outcome

$$\sum_{s=K-1}^{K-1} g^i(x_s, u_s, v_s) + g_K^i(x_K)$$

Find $\gamma_{K-1}^*, \sigma_{K-1}^*$ that are Nash equilibria for the pair of cost functions

$$g^1(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1})) + V_{1,K}(f(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1}))),$$
$$g^2(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1})) + V_{2,K}(f(x_{K-1}, \gamma_{K-1}(x_{K-1}), \sigma_{K-1}(x_{K-1})))$$

If the Nash equilibria above exist, plug them in and continue backwards. If at any iteration, a Nash equilibrium does not exist, the procedure fails.

- There may be other Nash equilibria for the game.

Backward induction in the one-player case

Bellman's principle of optimality: holds for the one-player case

Tail of an optimal policy is optimal for the tail subproblem.

State dynamics: $x_{k+1} = f(x_k, u_k)$
Finite-horizon cost function: $\sum_{k=0}^{K-1} g(x_k, u_k) + g_K(x_K)$
Tail subproblem: $\sum_{k=l}^{K-1} g(x_k, u_k) + g_K(x_K)$
Value function at stage k:

$$V_l(x) = \min_{(u_k, u_{k+1}, \ldots, u_{K-1})} \sum_{k=l}^{K-1} g(x_k, u_k) + g_K(x_K)$$

Value function and optimal policy satisfy the backward recursion:

$$\begin{aligned} V_K(x) &= g_K(x), \\ V_l(x) &= \min_{u \in \mathcal{U}} g(x, u) + V_{\ell+1}(f(x, u)),\ l = K-1, \ldots, 0, \\ \gamma_l^*(x) &\in \arg\min_{u \in \mathcal{U}} g(x, u) + V_{l+1}(f(x, u)),\ l = K-1, \ldots, 0. \end{aligned}$$

Single-player: the optimal control in state feedback policy $\gamma^* = (\gamma_0^*, \ldots, \gamma_{K-1}^*)$, is uniquely characterized by the above recursion and the optimal finite horizon cost $\min_{(u_0, \ldots, u_{K-1})} \sum_{k=0}^{K-1} g(x_k, u_k) + g_K(x_K)$ is $V_0(x_0)$ for any $x_0 \in \mathcal{X}$

One-player setting: Principle of Optimality

Tail of an optimal policy is optimal for the tail subproblem

Define the value functions V_l, $k = 0, 1, \ldots, K$ as follows.

$$V_K = g_K(x)$$

$$V_l(x) = \min_{(u_k, u_{k+1}, \ldots, u_{K-1})} \sum_{k=l}^{K-1} g(x_k, u_k) + g_K(x_K), \; \forall x \in \mathcal{X},$$

with the sequence $\{x_k\}_{k=l}^{K-1}\}$ starting at x, and satisfying the dynamics $x_{k+1} = f(x_k, u_k)$. Show that $V_l(x) = \min_{u \in \mathcal{U}} g(x, u) + V_{l+1}(f(x, u))$.

proof:

$$V_l(x) = \min_{(u_\ell, u_{\ell+1}, \ldots, u_{K-1})} \sum_{k=l}^{K-1} g(x_k, u_k) + g_K(x_K)$$

$$= \min_{u_\ell} \left[g(x_\ell, u_\ell) + \min_{(u_{\ell+1}, \ldots, u_{K-1})} \sum_{k=l+1}^{K-1} g(x_k, u_k) + g_K(x_K) \right]$$

$$= \min_{u_k} g(x_k, u_k) + V_{l+1}(f(x_k, u_k)).$$

Above, the first equality holds by the definition of V_l, the second from the fact that $g(x_k, u_k)$ does not depend on u_l $l > k$, and the last one holds by definition of V_{l+1} and the dynamics $x_{k+1} = f(x_k, u_k)$.

Infinite horizon settings

Need to ensure convergent series, one approach: geometrically discounting costs

- infinite horizon discounted cost

- Bellman equation

- Stationary policy

Computational implications

Consider having K stages and $|U_l|$ number of actions available at stage l.

1. Exhaustive search over all possible selections of actions requires comparing the costs associated with $|U_1| \times |U_2| \times \cdots \times |U_K|$ options.
2. Dynamic programming requires comparing for a specific value of the state x, requires comparing $|U_l|$ options for each state, in stage l. Thus, the total number of comparisons is $|U_1| \times |X_1| + |U_2| \times |X_2| + \cdots + |U_K| \times |X_K|$.

Cost-savings example

Hespanha - -

dummy Tic-Tac-Toe, where there is only one player.

1. Exhaustive search: $9! = 362880$.
2. Dynamic programming: 19107

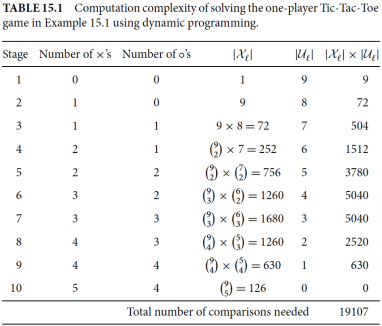
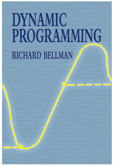


TABLE 15.1 Computation complexity of solving the one-player Tic-Tac-Toe game in Example 15.1 using dynamic programming.

Stage	Number of ×'s	Number of o's	$\lvert\mathcal{X}_\ell\rvert$	$\lvert\mathcal{U}_\ell\rvert$	$\lvert\mathcal{X}_\ell\rvert \times \lvert\mathcal{U}_\ell\rvert$
1	0	0	1	9	9
2	1	0	9	8	72
3	1	1	$9 \times 8 = 72$	7	504
4	2	1	$\binom{9}{2} \times 7 = 252$	6	1512
5	2	2	$\binom{9}{2} \times \binom{7}{2} = 756$	5	3780
6	3	2	$\binom{9}{3} \times \binom{6}{2} = 1260$	4	5040
7	3	3	$\binom{9}{3} \times \binom{6}{3} = 1680$	3	5040
8	4	3	$\binom{9}{4} \times \binom{5}{3} = 1260$	2	2520
9	4	4	$\binom{9}{4} \times \binom{5}{4} = 630$	1	630
10	5	4	$\binom{9}{5} = 126$	0	0
		Total number of comparisons needed			19107

Historical note

- Dynamic programming was developed by Richard Bellman in the 1950s
- It is the foundation of stochastic control and reinforcement learning
- Modern reference: Dynamic programming and optimal control by Dimitri Bertsekas



Summary and further reading

- Loop model: allows for a more general class of multi-stage games, and a more efficient computation of Nash equilibrium
- Backward induction can be used in games under perfect information: where both players know the state of the game at each stage
- The single-player setting corresponds to a deterministic optimal control problem
- The multi-player setting corresponds to a deterministic dynamic game
- The result can be extended to stochastic game setting. This setting will generalize stochastic optimal control.
- Reading: 141.1-14.~~4~~, 15.1-15.4 of Hespanha

  14.2