COMP3702/7702 Artificial Intelligence
Week 6: Decision making under Non-Deterministic and Adversarial Uncertainty
RN ch. 4.3, 5 & Prof. J.C. Latombe course CS121

Hanna Kurniawati
Topics until mid September

- What is AI.
- Search.
- Motion planning.
- Logic and its applications in action planning.
- Decision making under non-deterministic & adversarial uncertainty + Quantifying uncertainty.
- Decision making under stochastic uncertainty.
- Intro to Bayesian Inference.

How to make good decisions when the agent’s states are fully observed & the world dynamics are deterministic.
Topics until mid September

- What is AI.
- Search.
- Motion planning.
- Logic and its application in action planning.
- Decision making under non-deterministic & adversarial uncertainty + Quantifying uncertainty.
- Decision making under stochastic uncertainty.
- Intro to Bayesian Inference.

How to make good decisions when the agent’s states are fully observed & the world dynamics are deterministic.

How an agent can make good decisions when information is imperfect, i.e., real world...
Assumptions on environment

- **Fully observable** vs. **partially observable**.
  - Does the agent know the state of the world exactly?

- **Deterministic** vs. **non-deterministic**.
  - Does an action map one state into a single other state?

- **Static** vs. **dynamic**.
  - Can the world change while the agent is “thinking”?

- **Discrete** vs. **continuous**.
  - Are the actions & percepts discrete?

- **Known** vs. **unknown**.
  - Does the agent know the outcome(s) of performing an action from a state?
Assumptions on environment

- Fully observable vs. **partially observable**.
  - Does the agent know the state of the world exactly?

- Deterministic vs. **non-deterministic**.
  - Does an action map one state into a single other state?

- Static vs. **dynamic**.
  - Can the world change while the agent is “thinking”?

- **Discrete** vs. **continuous**.
  - Are the actions & percepts discrete?

- Known vs. **unknown**.
  - Does the agent know the outcome(s) of performing an action from a state?
Agenda

- Where uncertainty comes from?
- Planning in non deterministic world.
- Planning in adversarial world: Games.
- Quantifying uncertainty
Agenda

- Where uncertainty comes from?
- Planning in non deterministic world.
- Planning in adversarial world: Games.
- Quantifying uncertainty
Simple illustration: Assignment 1

- Model the problem
  - Parameterization (defining the C-space).

- Simplify the problem
  - Generating the state graph.

- Find the path from initial to goal.

- Feed the path to the ASVs.

- Will it work?
  - Most likely, no.
  - When the environment is relatively free of obstacles, the disturbances due to current is negligible, almost no disturbances from living organisms in the sea/air, the ASVs have great controller that errors are very small, and the oil behaves “nicely” -- always following the shape of the boom--, then maybe.
Don’t despair…

- There are real application…
  - Problems where we have control on the environment structure, e.g. car factory, assembly maintainability study, warehouse management systems (Kiva), etc.

- The methods you’ve studied are the basic building blocks for solving the real oil containment problem 😊.
Simple illustration: Assignment 1

- Model the problem
  - Parameterization (defining the C-space).
- Simplify the problem
  - Generating the state graph.
- Find the path from initial to goal.
- Feed the path to the ASVs.
- Will it work?
  - Most likely, no.
  - When the environment is relatively free of obstacles, the disturbances due to current is negligible, almost no disturbances from living organisms in the sea/air, the ASVs have great controller that errors are very small, and the oil behaves “nicely” -- always following the shape of the boom--, then maybe.
Causes of uncertainty:
System noise & errors

- Control error or disturbances from external forces
  - Effect of performing an action is non-deterministic.
- Errors in sensing & in processing of sensing data
  - Imperfect observation about the world (partially observable).
Causes of uncertainty: Modeling error

- Lazy.
  - Rolling dice in a casino. Depends on wind direction from air conditioning, number of people around the table,

- Reduce computational complexity.
  - Eliminate variables that will not affect the solution significantly.

- Accidental error.
Causes of uncertainty:
Problem simplification

- The actual possible states are often too large.
- Simplify, so it’s solvable by current computing power.
- But, in general simplification means clustering several actual states together and assume all actual state in the same cluster are the same.
  - Meaning: A state in our model corresponds to a set of actual states that are not differentiable by the program.
- Similarly with action space.
  - The effect of performing an action becomes non deterministic.
- Usually, bounded uncertainty.
Agenda

- Where uncertainty comes from?
- Planning in non deterministic world.
- Planning in adversarial world: Games.
- Quantifying uncertainty
Assumptions on class environment

- **Fully observable** vs. **partially observable**.
  - Does the agent know the state of the world exactly?

- **Deterministic** vs. **non-deterministic**.
  - Does an action map one state into a single other state?

- **Static** vs. **dynamic**.
  - Can the world change while the agent is “thinking”?

- **Discrete** vs. **continuous**.
  - Are the actions & percepts discrete?

- **Known** vs. **unknown**.
  - Does the agent know the outcome(s) of performing an action from a state?
Making decision

- We want to find a plan that works regardless of what outcomes actually occur.
  - Can no longer rely on a sequence of actions.
  - Need a conditional plan. The action to perform depends on the outcome of previous action.
AND/OR search tree

- A tree with interleaving AND and OR levels.
- At each node of an OR level, branching is introduced by the agent’s own choice.
- At each node of an AND level, branching is introduced by the environment.
Example: Slippery vacuum robot

- **States:** Conjunctions of
  - Robot in $R_1$, Robot in $R_2$,
  - $R_1$ clean, $R_2$ clean.

- **Action:**
  - Left, Right, Suck($R_1$), Suck($R_2$).

- **World dynamics:**
  - Non deterministic

- **Initial state:**
  - Robot in $R_1 \land R_1$ is clean.

- **Goal state:**
  - $R_1$ is clean $\land$ $R_2$ is clean.

After performing an action at a state, the robot may end up in one of several possible states.
Example: Slippery vacuum robot

- **World dynamics:**
  - \( \text{Succ}(\text{Robot in } R_1, \text{Right}) = \{\text{Robot in } R_1, \text{Robot in } R_2\} \).
  - \( \text{Succ}(\text{Robot in } R_2, \text{Right}) = \{\text{Robot in } R_1, \text{Robot in } R_2\} \).
AND/OR tree of slippery vacuum robot

**Right**
- Suck(R₁)

**State nodes** (agent decision nodes)

**Action nodes** (world "decision" nodes)

Do we have a solution?
AND/OR search tree

- Solution is a sub-tree that:
  - Has a goal node at every leaf.
  - Specifies one action at each node of an OR level.
  - Include every outcome branch at each node of an AND level.
Labeling an AND/OR Tree

- Assume no detection of revisited states.

OR level (state nodes)  ________________

AND level (action nodes)  ------

[Diagram of an AND/OR tree with labeled nodes and branches.]
Labeling an AND/OR Tree

- A leaf state node is **solved** if it’s a goal state.
- A leaf state node is **closed** if it has no successor and is not a goal.
Labeling an AND/OR Tree

- An action node is **solved** if all its children are solved.
- An action node is **closed** if at least one of its children is closed.
Labeling an AND/OR Tree

- A non-leaf state node is **solved** if one of its children is solved.
- A non-leaf state node is **closed** if all its children are **closed**.
Labeling an AND/OR Tree

- Keep labeling until the root.
Labeling an AND/OR Tree

- Keep labeling until the root.
Labeling an AND/OR Tree

- The problem is solved when the root node is solved
- The problem is impossible if the root node is closed
Solution of an AND/OR Tree

- The **solution** is the subtree that establishes that the root is solved.
Solution of an AND/OR Tree

- The **solution** is the sub-tree that establishes that the root is solved.

- It defines a **conditional plan** (or contingency plan) that includes tests on sensory data to pick the next action.

Conditional plan:
- If $s_1$ is observed then perform $a_2$
- Else if $s_2$ is observed then perform $a_3$
When a node is the same as an ancestor node

- Create a loop.
- Label?
  - Solved.
    Meaning: The solution is a conditional plan that includes loop.
    While (Robot in R₁) do Right
    Depends on the cause of non deterministic action, may / may not work.
  - Closed.
    Meaning: No solution through the node.
Search an AND/OR Tree

- Start from a state node (OR level).
  - Fringe nodes are state nodes.
- Use any search algorithms we have studied,
  - Select a fringe node to expand.
  - Select an action to use.
  - Insert the corresponding action node.
  - Insert all possible outcome of the action, as the child of the action node.
  - Backup to (re-)label the ancestor nodes.
- Cost calculation at AND level:
  - Weighted sum (when uncertainty is quantified using probability, expectation).
  - Take the minimum.
Agenda

- Where uncertainty comes from?
- Planning in non deterministic world.
- Planning in adversarial world: Games.
- Quantifying uncertainty
Adversarial world with examples in Games

- Making good decisions requires respecting your opponents.
  - Take into account what your opponents will do.
  - Assume your opponents are smart (at least until proven otherwise 😊).
A specific game

- Two-player.
- Turn-taking.
- Deterministic.
  - The game is deterministic. But the agent does not know what the opponent will do, and hence to the agent, the environment is non-deterministic.
- Fully observable.
- Zero-sum.
  - The total gain from the winning participants minus the total losses from the losing participants is zero.
  - Essentially, one’s winning means the opponent’s lost.
  - E.g. **tic-tac-toe**, chess, go.
Defining the problem

- State space
- Action space
- Initial state
- World dynamic: Represents the outcome of the agent’s move, followed by the possible game state after the opponent moves.
- Utility: +1 win, -1 lose.
Search tree called Game tree

- Similar to AND/OR tree.
- OR level: The agent’s move.
  - Maximize value.
- AND level: The opponent’s move.
  - Minimize value.
Game tree

Agent’s move (MAX) →

Opponent’s Move (MIN) →

Terminal state (win for MAX) →
In general, the branching factor and the depth of terminal states are large

Chess:
• Number of states: \( \sim 10^{40} \)
• Branching factor: \( \sim 35 \)
• Number of total moves in a game: \( \sim 100 \)
Online search

- Calculate the solution for the current state.
  - Perform a search with current state as root.
  - Perform the first action of the solution.
  - Recompute the solution from the new state, utilizing the previous sub-trees if possible.
Choosing an Action: Basic idea of Minimax Algorithm

- Using the current state as the initial state, build the game tree to the maximal depth $h$ (called horizon) feasible within the time limit.

- Evaluate the states of the leaf nodes.
  - Use heuristic as an evaluation function to estimate how favorable a state is.

- Back up the results from the leaves to the root and pick the best action assuming the worst from MIN.
  - At each non-leaf node $N$, the backed-up value is the value of the best state that MAX can reach at depth $h$ if MIN plays well (by the same criterion as MAX applies to itself).
  - Same criterion: same evaluation function.
Evaluation function

- Need a heuristic to estimate how favorable is a game state for the agent (MAX).
  - Usually called evaluation function $e: S \rightarrow R$.
  - $e(s) > 0$: $s$ is favorable to MAX (the larger the better).
  - $e(s) < 0$: $s$ is favorable to MIN.
  - $e(s) = 0$: $s$ is neutral.
Example: Tic-tac-toe

- $e(s) =$ number of rows, columns, and diagonals where MAX can win - number of rows, columns, and diagonals where MIN can win.
- Agent (MAX): cross.

- $8-8 = 0$
- $6-4 = 2$
- $3-3 = 0$
Construction of an Evaluation Function

- Usually a weighted sum of “features”:
  \[
e(s) = \sum_{i=1}^{n} w_i f_i(s)\]
  
  \(w\): weight.
  
  \(f(s)\): features.

- Features may include
  - Number of pieces of each type
  - Number of possible moves
  - Number of squares controlled
Tic-Tac-Toe tree at horizon = 2

Example: Backing up Values

Best move
Minimax Algorithm

1. Expand the game tree from the current state (where it is MAX’s turn to play) to depth \( h \)
2. Compute the evaluation function at every leaf of the tree
3. Back-up the values from the leaves to the root of the tree as follows:
   a. A MAX node gets the maximum of the evaluation of its successors
   b. A MIN node gets the minimum of the evaluation of its successors
4. Select the move toward a MIN node that has the largest backed-up value
Game Playing (for MAX)

Repeat until a terminal state is reached
1. Select move using Minimax
2. Execute move
3. Observe MIN’s move

Note that at each cycle the large game tree built to horizon $h$ is used to select only one move
All is repeated again at the next cycle (a sub-tree of depth $h-2$ can be re-used)
Can we do better?

Yes! Much better!

This part of the tree can’t have any effect on the value that will be backed up to the root.
Idea of Alpha-Beta Pruning

- $\alpha$: Best already explored option along path to the root for maximizer.
- $\beta$: Best already explored option along path to the root for minimizer.
- Explore the game tree to depth $h$ in depth-first manner.
- Back up $\alpha$ and $\beta$ values whenever possible.
- Prune branches that can’t lead to changing the final decision.
Example
The beta value of a MIN node is an upper bound on the final backed-up value. It can never increase.
The beta value of a MIN node is an upper bound on the final backed-up value. It can never increase.
Example

The alpha value of a MAX node is a **lower** bound on the final backed-up value. It can never decrease.

\[ \alpha = 1 \]

\[ \beta = 1 \]
Example

\[ \alpha = 1 \]

\[ \beta = 1 \]

\[ \beta = -1 \]
Example

\[ \alpha = 1 \]

Search can be discontinued below any MIN node whose beta value is less than or equal to the alpha value of one of its MAX ancestors.
Alpha-Beta Algorithm: When to Prune?

- Update the alpha/beta value of the parent of a node N when the search below N has been completed or discontinued.
- Discontinue the search below a MAX node N if its alpha value is $\geq$ the beta value of a MIN ancestor of N.
- Discontinue the search below a MIN node N if its beta value is $\leq$ the alpha value of a MAX ancestor of N.
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
Example
How much do we gain?

Consider these two cases:

Node ordering matters for efficiency!
How much do we gain?

- Assume a game tree of uniform branching factor $b$
- Minimax examines $O(b^h)$ nodes, so does alpha-beta in the worst-case
How much do we gain?

- The gain for alpha-beta is maximum when:
  - The MIN children of a MAX node are ordered in decreasing backed up values
  - The MAX children of a MIN node are ordered in increasing backed up values

- Then alpha-beta examines $O(b^{h/2})$ nodes [Knuth and Moore, 1975]

- But this requires an oracle (if we knew how to order nodes perfectly, we would not need to search the game tree)

- If nodes are ordered at random, then the average number of nodes examined by alpha-beta is $\sim O(b^{3h/4})$
Heuristic Ordering of Nodes

- Order the children of a node according to the values backed-up at the previous iteration
Computer programs have beaten some the best human players

- 1994: Chinook beats Mr. Tinsley in Checkers.
  - Mr. Tinsley is world champion of checkers for over 40 years.
  - Try it: http://webdocs.cs.ualberta.ca/~chinook/

- 1997: Deep blue beats Mr. Kasparov.
  - Mr. Kasparov is world champion in chess during 1985-2000.
How they (the comp. game makers) did it?

- Many game programs are based on alpha-beta + iterative deepening + extended/singular search + transposition tables + huge databases + …
- The methods are general, but their implementation is dramatically improved by many specifically tuned-up enhancements (e.g., the evaluation functions).
Agenda

- Where uncertainty comes from?
- Planning in non deterministic world.
- Planning in adversarial world: Games.
- Quantifying uncertainty
FATHER (F): Nurse, what is the probability that the drug will work?

NURSE (N): I hope it works, we’ll know tomorrow.

F: Yes, but what is the probability that it will?

N: Each case is different, we have to wait.

F: But let’s see, out of a hundred patients that are treated under similar conditions, how many times would you expect it to work?

N (somewhat annoyed): I told you, every person is different, for some it works, for some it doesn’t.

F (insisting): Then tell me, if you had to bet whether it will work or not, which side of the bet would you take?

N (cheering up for a moment): I’d bet it will work.

F (somewhat relieved): OK, now, would you be willing to lose two dollars if it doesn’t work, and gain one dollar if it does?

N (exasperated): What a sick thought! You are wasting my time!
Probabilistic Modeling

View:
- Experiments with random outcome.
- Quantifiable properties of the outcome.

Three components:
- Sample space: Set of all possible outcomes.
- Events: Subsets of sample space.
- Probability: Quantify how likely an event occurs.
Probability

- Probability: A function that maps events to real numbers satisfying these axioms:
  1. Non-negativity: $P(E) \geq 0$, where $E$ is an event.
  2. Normalization: $P(S) = 1$, where $S$ is the sample space.
  3. Additivity of finite / countably infinite events.

$$P\left(\bigcup_{i=1}^{\infty/n} E_i\right) = \sum_{i=1}^{\infty/n} P(E_i),$$

where $E_i$ are disjoint / mutually exclusive, $i: \text{natural number}$. 
Use of Probability Axioms

Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations. Which is more probable?

1. Linda is a bank teller.
2. Linda is a bank teller and is active in the feminist movement.

Tversky & Kahneman.
Conditional Probability

- Sometimes, knowing something makes a difference.
- Model/reason about the outcome of an experiment, based on partial information.
- Given that event A occurs.
- The probability that event B also occurs:

\[ P(B \mid A) = \frac{P(A \cap B)}{P(A)} \]

- Extendable to knowing multiple events.
Chain Rule

- Probability that two events occur:

\[ P(A \cap B) = P(B \mid A)P(A) \]

- In general,

\[
P\left(\bigcap_{i=1}^{n} A_i\right) = P\left(A_1 \bigcap_{j=2}^{n} A_j\right)P\left(\bigcap_{j=2}^{n} A_j\right) = P\left(A_1 \bigcap_{j=2}^{n} A_j\right)P\left(A_2 \bigcap_{k=3}^{n} A_k\right)\ldots P\left(A_{n-1} \mid A_n\right)P(A_n)
\]
Bayes Rule

- Knowing $P(A|B)$ maybe easier than knowing $P(B|A)$, e.g.,
  - Knowing symptoms of a disease is easier than figuring out the disease given the symptoms.
Example

- A cab was involved in a hit & run accident at night.
- Only 2 cabs company operate in the city, the Blue & the Green.
- 85% of the cabs in the city are Green.
- The court tested the reliability of the witness under the same circumstances that existed on the night of the evidence & concluded that the witness correctly identify the color of the taxi 80% of the time.
- A witness identified the cab is Blue.
- Is Blue cab more likely to be the one involved in the accident ?

Tversky & Kahneman.
B : Blue cab was involved in the accident.
G : Green cab was involved in the accident.

\( W_B \) : The witness says blue is the one involved in the accident.

\( W_G \) : The witness says green is the one involved in the accident.

\( P(B) = 0.15 \quad ; \quad P(G) = 0.85 \)

\( P(W_B | B) = P(W_G | G) = 0.8 \quad ; \quad P(W_B | G) = P(W_G | B) = 0.2 \)

\[
P(B | W_B) = \frac{P(W_B | B)P(B)}{P(W_B)} = \frac{P(W_B | B)P(B)}{P(W_B \cap B) + P(W_B \cap G)}
\]

The same derivation applies to \( P(G | W_B) \).

The answer to the question is blue if \( P(B | W_B) > P(G | W_B) \) and vice versa.
Independence

- When knowing something doesn’t make a difference.
- Knowing event A occurs does not change the probability that event B occurs:
  \[ P(B|A) = P(B) \rightarrow P(A \cap B) = P(A)P(B) \]
- Chain rule becomes:
  \[ P\left( \bigcap_{i=1}^{n} A_i \right) = \prod_{i=1}^{n} P(A_i) \]
Random Variables

- Interest is on numerical values associated with samples, e.g.:
  - Sample 50 students enrolled in COMP3702/7702, the number of students from mechatronics, IT, s/w eng., bioinf.
  - Roll a fair dice, get $5 if the outcome is even, & lose $5 if the outcome is odd.

- Random variable $X$ is a function $\mathbf{X} : S \rightarrow \text{Num}$.
  - Num: countable set (e.g., integer) $\rightarrow$ discrete random variable.
  - Num: uncountable set (e.g., real) $\rightarrow$ continuous random variable.
Characterizing Random Variables

- Cumulative distribution function (cdf)
  \[ F_X(x) = P(X \leq x) = P\left(\{s|X(s) \leq x, s \in S\}\right) \]

- Discrete: Probability mass function (pmf)
  \[ f_X[x] = P(X = x) \]

- Continuous: Probability density function/probability distribution function (pdf)
  \[ f_X(x) = \frac{dF_X(x)}{dx}; \quad P(a \leq X \leq b) = \int_a^b f_X(x)dx \]
More Compact Characterization of Random Variables

- Expectation: Weighted average of possible values of X, weight: probability.

\[
E[X] = \sum_x xf_X(x) \quad ; \quad E[X] = \int_{-\infty}^{\infty} xf_X(x) dx
\]

\[
E[g(X)] = \sum_x g(x)f_X(x) \quad ; \quad E[g(X)] = \int_{-\infty}^{\infty} g(X)f_X(x) dx
\]

Linearity of Expectation: \(E[aX + b] = aE[X] + b\)
More Compact Characterization of Random Variables

- **Variance**: A measure of dispersion around the mean.
  \[
  \text{var}(X) = E\left[\left(X - E[X]\right)^2\right] = E[X^2] - \left(E[X]\right)^2
  \]

- **Standard deviation**:
  \[
  \sigma(X) = \sqrt{\text{var}(X)}
  \]

When \( g \) is linear:
\[
\text{var}(aX + b) = a^2 \text{ var}(X)
\]
For more on probability...

- STAT2202/STAT2203.
- ENGG7302.
- Introduction to Probability by Bertsekas & Tsitsiklis.