

Computing Science (CMPUT) 455

Search, Knowledge, and Simulations

Ting-Han Wei

Department of Computing Science
University of Alberta
`tinghan@ualberta.ca`

Fall 2020

455 Today - Lecture 5

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

Today's Topics:

- More on size of state space, effort of solving a game
- Sequential decision-making

Coursework and New Uploads

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Quiz 3
- Assignment 1
- Reading O'Neil, How algorithms rule our working lives
- Activities Lecture 5
- Python codes `count_dag.py`,
`generate_tree.py`, `generate_tree_test.py`

Activities 5a and 5b

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Compute size of game trees
- When does a tree grow more quickly?
 - When increasing b ?
 - When increasing d ?
- Compare two cases

Size and Structure of State Space for Games

Review: State Space vs Playing and Solving a Game

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- What is the complexity of solving, or playing well, in a game?
- Depends on many factors:
 - Branching factor
 - Depth
 - Existence of a simple strategy
 - Existence of a mathematical theory
 - Having master players, master games, books to learn from
 - Having good heuristics
 - ...

Review: State Space vs Playing and Solving a Game

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

Simple measures of complexity of state space

- Size: how many states?
- Structure: tree, DAG, or DCG?
- Branching factor b
- Depth d

Estimating Search Effort

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Assume we need to visit every state in order to solve a game
 - (Later we will see we can do much better)
- How long does it take?
- Main factors:
 - Speed of program
 - Size of state space
- Let's look at 7×7 Go and Go1

7 × 7 Go - Example

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- 7 × 7 Go, start on empty board
- Assume we can process 1000 states/second
- Assume simplest tree model, $b = 49$
- What depth d can we reach in which time?
- Can we explore the whole state space?

Brute Force Search for 7×7 Go

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

Table: Each row shows the estimated *additional effort* to search one level deeper

Depth	New states	Added search time
0	1	1ms
1	49	50ms
2	49^2	2.4s
3	49^3	2 min
4	49^4	1.6 hrs
5	49^5	3.2 days
6	49^6	160 days
7	49^7	21.5 years
...

Fighting Exponential Growth

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- We cannot even search 7 moves deep with Go1
- To solve the game we need to see to the end
- This can be over 30 moves deep even for this small board
- Time limits in practice
 - 5 sec - 5 min per move in a tournament game
 - Maybe a few months to solve a game
- How can we succeed?
 - Increase speed of program (Lecture 6)
 - Decrease branching factor b (now)
 - Decrease depth d (later)

Decrease Branching Factor b

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

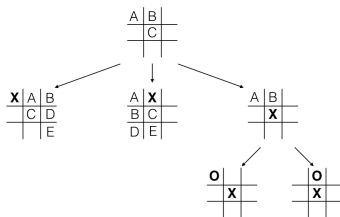
- Branching factor = growth of number of states per level
- How to decrease?
 - Reduce number of moves (but how?)
 - Use DAG instead of tree
 - Better search algorithms (e.g. alphabeta search)
- First try: take symmetry into account

Example - Use Symmetry in TicTacToe

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making



- At root: Only 3 of 9 total moves are different
 - Corner (A), Edge (B), Center (C)
- All 6 other moves lead to a symmetric position, same result as A or B
- Symmetries at tree level 1:
 - After corner or edge move: 5 distinct cases
 - After center move: 2 distinct cases
- Limitation: most symmetries broken after few moves

Symmetry

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

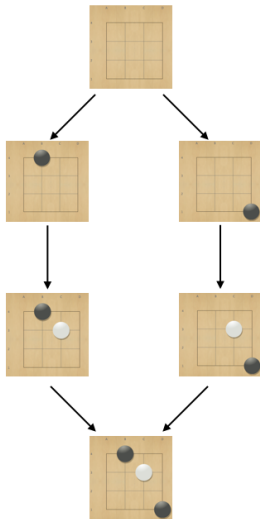
- Typical example to reduce state space by symmetry
- Good reduction at depth 1 or 2
- Then *symmetry breaks*
- Almost no reduction deeper in the tree
- Reduction of whole state space is limited to some constant factor
 - Less than 8 in Go

DAG (Directed Acyclic Graph)

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making



- Idea: single node for all *equivalent* states
- Different paths to same node
- Can lead to huge reduction in state space
- Why?
 - The whole *subtree* below is no longer duplicated
 - Can happen throughout the whole game

Tree vs DAG

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Tree model
 - Each action leads to a new node
- DAG model
 - All *equivalent* states represented by a single node
- Reduction in size of state space
 - Can be many orders of magnitude
 - Examples: Activities 5c - 5e
- Advantages of DAG model:
- Avoid redundant computations
 - No copied subtrees or sub-DAGs
- Share results of analysis - compute once, re-use often

Limitations of DAG Model

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Main problems:
 - Need memory to store and recognize equivalent states
 - Some algorithms designed only for trees, not for DAGs
- Example: propagating information up towards root
 - Only one path up in tree - efficient
 - Many paths in DAG - many ancestors

More Limitations of DAG Model

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

Limitation: states with different *history*

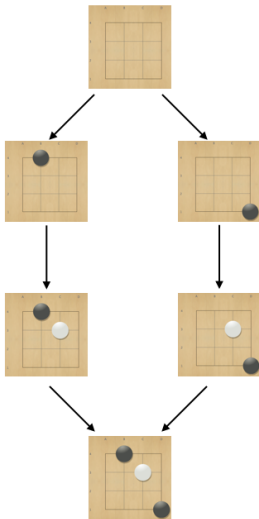
- Cannot always merge into one node, not always equivalent
- Example: simple ko - is capture allowed?
- Board looks the same but moves are different
- Can you still do something? Yes. One of Martin's students wrote a whole PhD thesis on such questions (Kishimoto 2005)

Counting States in a DAG

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making



- Simplified NoGo example, also counts illegal states
- Depth 0: 0 black, 0 white stones
- Depth 1: 1 black, 0 white stones
- Depth 2: 1 black, 1 white stones
- Depth 3: 2 black, 1 white stones
- Depth d : $\lceil d/2 \rceil$ black stones and $\lfloor d/2 \rfloor$ white stones
- How many ways to put that many stones on a board with 49 points?

Counting States in a DAG (continued)

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Example:
- Board with 49 squares
- How many different ways to place 5 black stones?
- Answer: $\binom{49}{5} = 1906884$
- Need to review the math background? See e.g.
<https://en.wikipedia.org/wiki/Combination>

Counting States in a DAG (continued)

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- How many different ways to place 5 black stones and 3 white stones?
- Answer: $\binom{49}{5} \times \binom{44}{3} = 1906884 \times 13244 \approx 25.2$ billion
- Why?
 - $\binom{49}{5}$ ways to place 5 black stones
 - $49 - 5 = 44$ empty points remaining
 - $\binom{44}{3}$ ways to place the 3 white stones there
 - Each different choice for either black or white leads to a different position, so multiply

Activity 5d: Count States in Tic-Tac-Toe DAG

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- TicTacToe board has 9 squares
- Ignore symmetry, early wins for now
- At level d : $\lceil d/2 \rceil$ X's and $\lfloor d/2 \rfloor$ O's
- Compute number of positions with 0, 1, 2, 3, ..., 9 stones

0 stones	0 X, 0 O	1 position
1 stone	1 X, 0 O	9 positions
2 stones	1 X, 1 O	? positions
3 stones	2 X, 1 O	? positions
4 stones	2 X, 2 O	? positions
...		? positions
9 stones	5 X, 4 O	? positions

Hint: Python code in `count_dag.py` is useful...

b^d Model and DAG

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Counted the number of states in a DAG at each depth d
- What about the branching factor?
- No longer a constant b for whole DAG
- Different *effective branching factors* b_d depending for each depth (or level) d
- Can compute b_d at depth d as:

$$b_d = \frac{\text{\#nodes-at-depth } d + 1}{\text{\#nodes-at-depth } d}$$

- Activity 5e: compute the effective branching factor for the TicTacToe DAG

Computing Size of State Space in DAG from Branching Factors

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Given branching factors, how many nodes in a DAG?
- 1 root node at depth 0
- b_0 children of root $\rightarrow b_0$ total nodes at depth 1
- Each child has b_1 new children
 \rightarrow total $b_0 \times b_1$ nodes at depth 2
- Depth n :
 $b_0 \times b_1 \times \dots \times b_{n-1}$ nodes

Computing Size from Branching Factors (continued)

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Total nodes up to depth d

1

+ b_0

+ $b_0 \times b_1$

+ ...

+ $b_0 \times b_1 \times \dots \times b_{d-1}$

- In general:

no nice closed-form solution for this sum

- In practice:

estimate branching factors b_i

- Use search or sampling
- Difficult or impossible to compute them exactly for large games

b^d Model vs Reality: Some Case Studies (1)

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- How realistic is the b^d model for size of state space?
- We'll look at some popular games
 - Go, TicTacToe:
 - Roughly, $b_n \approx b_0 - n$
 - Why? One less empty square with each move
 - One less possibility for next move
- Not exact:
 - Ignores illegal move rules
 - Ignores games that end earlier
- Setting $b_n = b_0 - n$ gives $b_0!$ leaf nodes
 - Earlier TicTacToe example: $9 \times 8 \times \dots 1$

b^d Model vs New estimate: 7×7 Go

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- 7×7 Go estimate from Lecture 4:
- 25 moves on average during a game, game length about 30 moves
- Rough b^d estimate $25^{30} \approx 10^{42}$
- New model:
- $b_0 = 49$, and $b_n \approx b_0 - n$
- Stop game at $n = 30$
- $49 \times 48 \times \dots \times (49 - 30) = 49!/18! \approx 10^{47}$
- Still ignores captures, ko, different game lengths,...

b^d Model vs Reality: Checkers

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

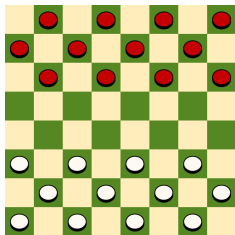


Image source:

<https://en.wikipedia.org>

- Checkers: complicated b
- Beginning: many pieces blocked
- Pieces unblocked: b increases
- Forced captures: $b = 1$
- When pieces get captured, b decreases again
- When checkers become kings, b strongly increases
- Estimated average over typical game: $b \approx 2.8$
- Length of game d varies wildly

b^d Model vs Reality: Chess

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

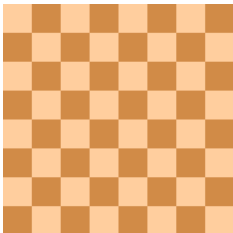


Image source:

<https://en.wikipedia.org>

- Chess: also complicated
- Pieces such as queens can have many moves, but may be blocked
- King in check: often only few legal moves
- When pieces get captured, b decreases
- Estimated average over typical game: $b \approx 35$
- Length of game d varies wildly

b^d Model vs Reality: Shogi

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

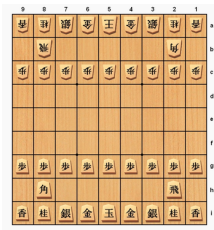


Image source:

<https://en.wikipedia.org>

- Shogi, Japanese chess
- Similar to chess, plus:
- Captured opponent pieces can be *reused* for yourself in a future move
- With captures, b *increases*
- Estimated average over typical game: $b \approx 92$
- b can be several hundred in endgame with many captured pieces available for “dropping” back on board

Complexity of Popular Games

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Big table in

`https:`

`//en.wikipedia.org/wiki/Game_complexity`

- Different measures of complexity
- Complexity also depends strongly on size of board
- In Go, the theoretical complexity is much higher
 - Main reason: capture, play again on same point
 - Example: Go player fills eyes, games last VERY long
 - Game ends only if **all** moves cause full-board repetition

Example: Solving 2×2 Go, $1 \times n$ Go

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- 2×2 Example from John Tromp,
`https://tromp.github.io/java/go/twoxtwo.html`
- Naive brute force minimax search: trillions of nodes
- Alphabeta with bad move ordering:
 - 19397529 nodes, max. depth 58
- Alphabeta with good move ordering:
 - 1446 nodes, max. depth 22
- Solving $1 \times n$ Go: *Exploring Positional Linear Go*
recent paper by Noah Weninger (UofA undergrad!) and
Ryan Hayward (UofA prof) - see resources

Sequential Decision-Making

Sequential Decision-Making

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

Topics:

- From decision making to sequential decision making
- Notation for action sequences
- View game tree as tree of move sequences

From State Space to Decision-Making

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- We studied state spaces in some detail
- Now, how do we find good actions (moves) in a state?
- In general, looking at the current state is not enough
- We need to look ahead to future states in order to make a good decision now
- We need to consider *sequences* of actions, until we reach a terminal state
- In games, each sequence is one possible way of playing the game

Making Complex Decisions

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- How to make good decisions?
- Consider many alternatives
- Consider short-term and long-term consequences
- Evaluate different options and choose the best-looking one
- Understanding and comparing sequences of actions is the main step in making such decisions

Making Sequential Decisions

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

Very general model:

Loop:

- Get current state of world
- Analyze it
- Select an action
- Observe the world's response
- If not done: go back to start of loop

Practically important question:

- Can we do this in a *simulation model* as opposed to the real world?

Single Agent Example: Path Planning

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

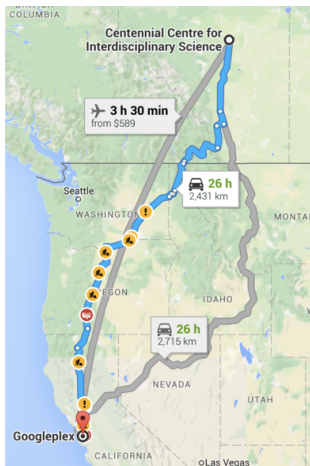


Image source: Googlemaps

- Task: start from here, visit Google headquarters
- First decision: fly, drive, take the bus, or walk?
- If drive or walk: each street corner is a decision point
- Need a long sequence of decisions to arrive at destination
- Is it *optimal*? Is it *good enough*?
- Tradeoffs: Speed vs cost vs scenery vs construction sites ...

Formal Framework

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Sequence of states and actions
- Start state s_0
- Action a_i leads to next state, s_{i+1}
- Keep going until reach a terminal state s_n
- Sequence $(s_0, a_0, s_1, a_1, \dots, s_n)$
- Sometimes we only write the actions (a_0, a_1, \dots, a_n)
 - Example: games where states are determined from game rules and actions

Formal Framework with Rewards

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Formal framework can also include rewards (or costs)
- Simple case (most games we consider):
single reward r at end
- General case: reward r_i after each action a_i
 - Write rewards as part of sequence:
 - $(s_0, a_0, r_1, s_1, a_1, r_2, \dots, r_n, s_n)$

Partial Sequences

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Full sequence goes all the way to terminal state
- *Partial sequence* can stop after any number of actions
- Two full sequences always share a *common prefix*
- In worst case, it might only contain the start state
 - Example - Go game
 - $(s_0, \text{Black B3}, s_1, \text{White A2}, s_{2a}, \text{Black D4})$
 - $(s_0, \text{Black B3}, s_1, \text{White A4}, s_{2b}, \text{Black D4})$
 - Common prefix $(s_0, \text{Black B3}, s_1)$

Re-Interpreting the Tree and DAG Models

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Our model so far:
- State space as a graph
- Nodes are states, edges are actions
- Tree and DAG are special cases of graphs
- New view:
we can view the same trees and DAGs
as a way to organize all action sequences

Organizing Sequences in Trees

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

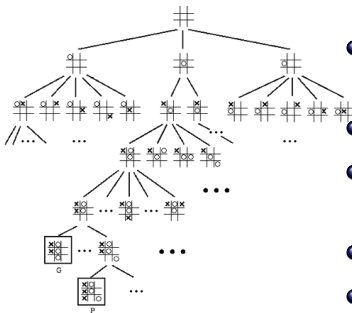


Image source: <http://web.emn.fr>

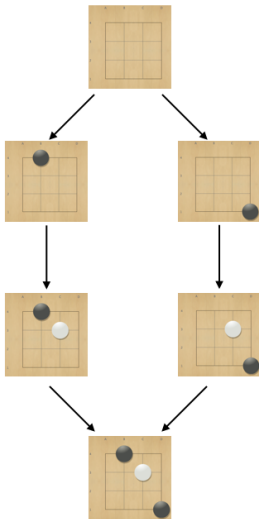
- Consider the (huge) set of all possible state-action sequences
- Organize them such that:
- Any two sequences share their *longest common prefix*
- Branch as soon as they differ
- Result: we get exactly the tree representation of the state space

Organizing Sequences in a DAG

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making



- Similarly, we can relate sequences to the DAG model
- Start with sequences-as-tree model
- Then, merge two different sequences when they both reach equivalent states
- Result: we get exactly the DAG representation

Summary

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Looked at more details and examples of state spaces
- Estimating size of state space in DAG model
- Theory vs reality: state space of some popular games
- Sequential decision-making model
- Relation between tree and DAG models, sequences of decisions

Preview - What's Next?

CMPUT 455

Size and
Structure of
State Space
for Games

Sequential
Decision-
Making

- Lecture 6: profiling and optimization
- Topics for next few weeks:
- Do we need to look at *all* possible sequences to make a decision?
- How do we use the tree or DAG structure?
- Algorithms for decision-making in games on tree and DAG structures
- How do we use heuristics?
- Simulation using random sequences of actions