CSCE 580: Introduction to AI

Week 9 - Lectures 16 and 17: Informed Search, Local Search

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

14TH OCT AND 16TH OCT 2025

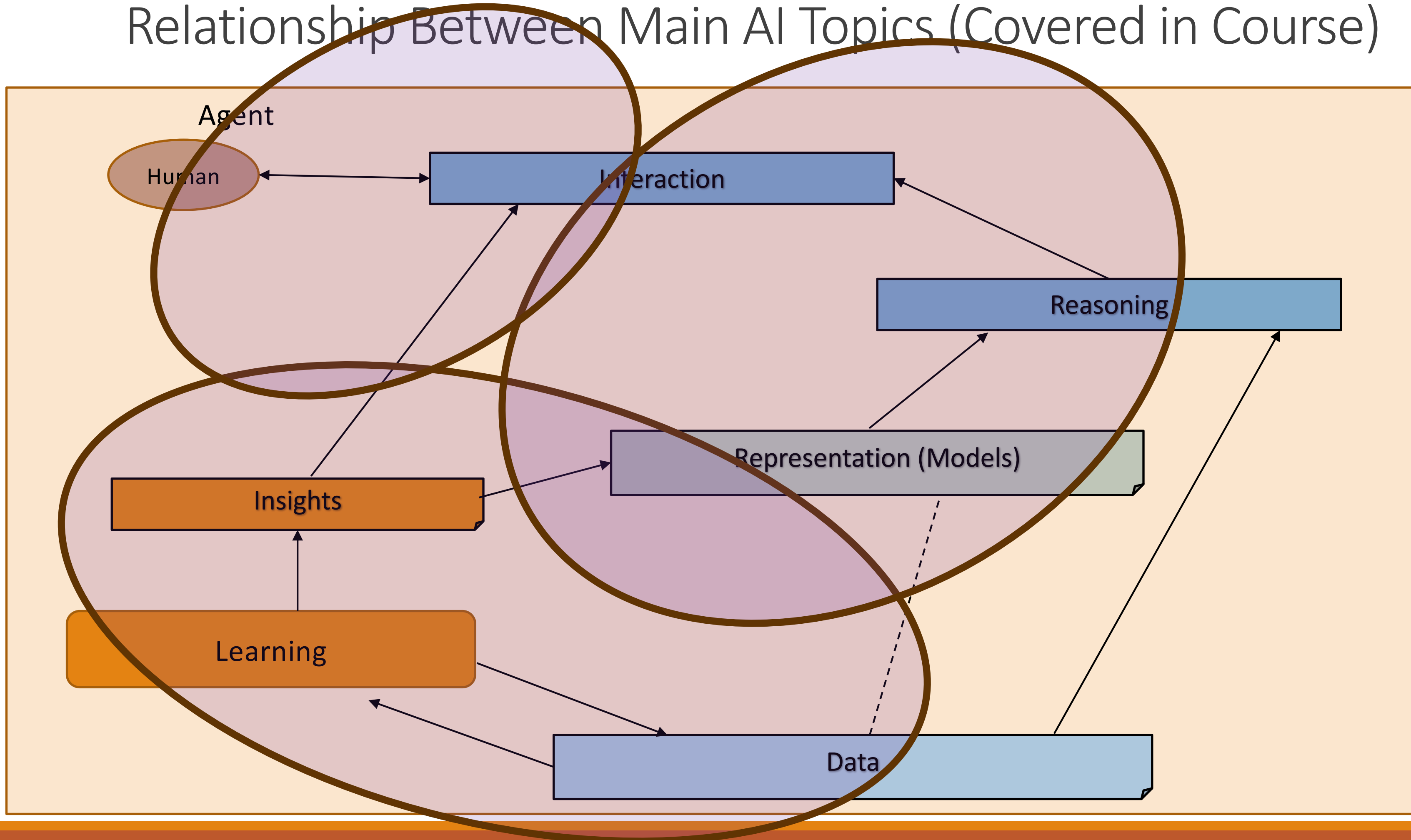
Carolinian Creed: "I will practice personal and academic integrity."

Credits: **Copyrights of all material reused acknowledged**

Organization of Week 9 - Lectures 16, 17

- Introduction Section
  - Recap from Week 5 (Lectures 9 and 10)
  - AI news
- Main Section
  - Lecture 16: Informed Search
    - Heuristic search
    - Optimal solutions
  - Lecture 17: Local Search
- Concluding Section
  - About next week – W10: Lectures 18, 19
  - Ask me anything

Relationship Between Main AI Topics (Covered in Course)



Recap of Week 8

We discussed

- Quiz 2
- Fall Break

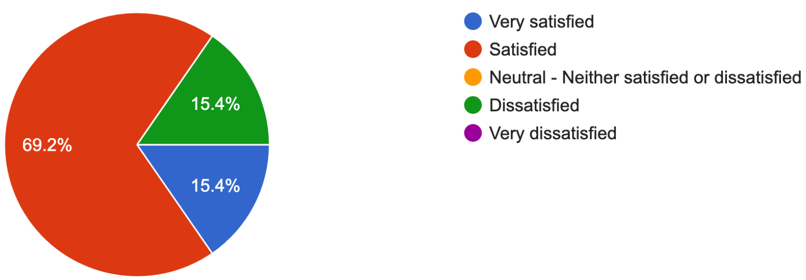
- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Upcoming Evaluation Milestones

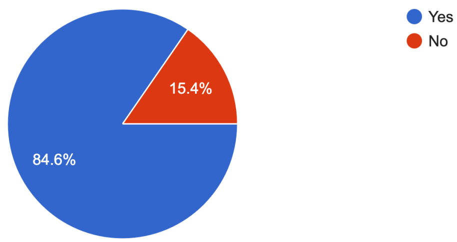
- Projects B: Sep 30 – Nov 20
- Quiz 2: Oct 7
- Quiz 3: Nov 11
- Paper presentation (grad students only) : Nov 18
- Finals: Dec 11

Mid-Course Survey

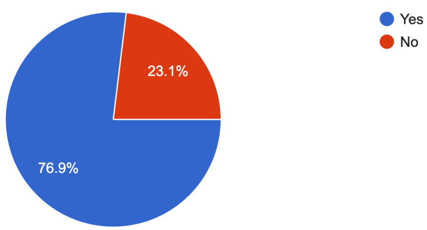
How satisfied are you with the course?
13 responses



Do you like the pace of the course ?
13 responses



Do you like the content on which the course is focusing?
13 responses



Mid-Course Survey - Pointers

- "the **use case in Quiz** 2 (converting a webpage to semi-structured data) **actually seemed useful**. It's one of the first times I've seen an LLM used for something that I think is objective useful, and for a task that can't (to my knowledge) be achieved with traditional programming methods"
- "The **mathematical details** of machine learning were **<u>not</u>** adequately covered"
- "I think it would be helpful to do more in **class lab-like assignments**, where we actually do some programming … "

Message: Go in-depth in a few topics!

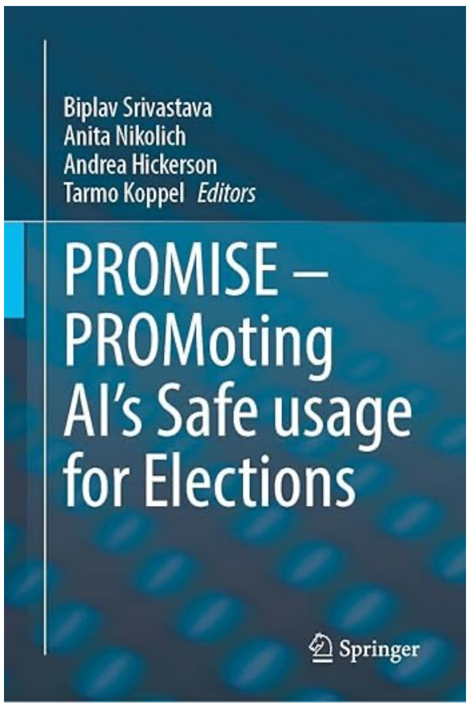
Mid-Course Survey – Changes Made

- Reiterate **in-depth topics**
  - ML methods – classification, explanation (done)
  - Search (ongoing)
  - Decision making – simple, complex decision making
- Highlight related courses
  - ML details: ML Systems (CSCE), Statistical ML (Maths Dept)
  - Trusted AI (CSCE 581; Spring 2026)
- Encourage exploration
  - Project B
  - Paper presentations (graduate students)

AI News

#1 NEWS – PROMISE Book - PROMoting AI's Safe usage for Elections

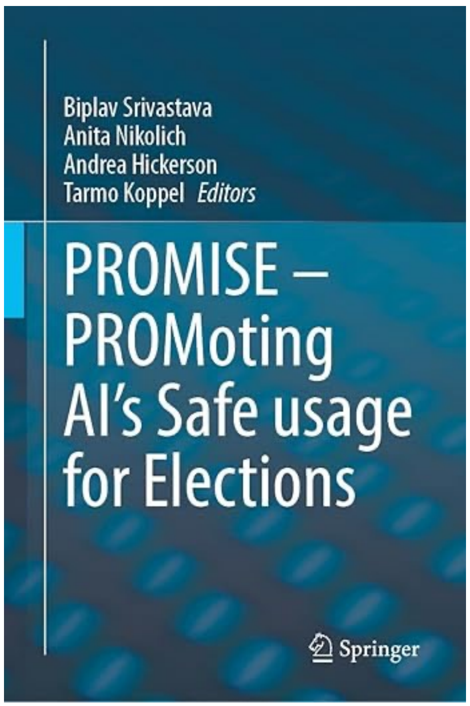
- Link: https://www.linkedin.com/posts/biplav-srivastava_promise-promoting-ais-safe-usage-for-elections-activity-7377405499815911424-nlgU
- Press: https://www.nbcnews.com/tech/security/hacker-used-ai-automate-unprecedented-cybercrime-spree-anthropic-says-rcna227309



- It is an edited book based on four years of work that explores AI's role in enhancing credible, data-driven electoral processes.
- This all started with a few of us brainstorming on how AI could make elections better that lead to academic workshops at Neurips 2021, AAAI 2023, AAAI 2024, a special issue of the AI Magazine on the topic, and a blue-sky paper at AAAI 2025.
- The book has contributions from over 30 authors that are drawn from academia, industry, non-profits, and government, from around the world. They bring and combine technical perspectives from the lens of computer science and AI, security, journalism, law, and political science, and consider elections in all continents - Asia (India), Africa (Ghana, Nigeria, Kenya), Europe (Estonia, UK), North America (Canada, US) and Latin America (Brazil).
- The book consists of a mix of article types - research papers, interviews and essays, touching on impact potentials of AI technologies like chatbots, large language models, game theory and machine learning, for voters, candidates, and election commissions, and **ends with a code of ethics for those working in AI and election space using relevant guidance from computing and journalism fields**.
- It offers practical guidelines for researchers, teachers, practitioners, students and government officials.

#1 NEWS – PROMISE Book - PROMoting AI's Safe usage for Elections

- Link: https://www.linkedin.com/posts/biplav-srivastava_promise-promoting-ais-safe-usage-for-elections-activity-7377405499815911424-nlgU
- Press: https://www.nbcnews.com/tech/security/hacker-used-ai-automate-unprecedented-cybercrime-spree-anthropic-says-rcna227309



Technology in Elections: Code of Ethics – The 7 Easy Reckoner

Promoting Computing: (ACM – https://www.acm.org/code-of-ethics)
•Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing
•Maintain high standards of professional competence, conduct, and ethical practice

Promoting Communication: (https://www.spj.org/ethicscode.asp)
•Seek truth and report it
•Be accountable and transparent

Promoting Model Citizenship Responsibility
•Minimize harm
•Respect everyone's view and give them space to express them
•Honor people and their free will to vote

#2 NEWS – Insights for Irmo Fire

- Based on results from their data and Quiz1 analysis

Introduction Section

Main Section

Lecture 16:
Informed Search

Uninformed Search Strategies

Search strategies use only the information available in the problem definition. They do not use a measure of distance to goal (*uninformed*).

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening search
- Bidirectional search

Consideration: type of queue used for the fringe of the search tree (collection of tree nodes that have been generated but not yet expanded)

Adapted from:
1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

Analyzing Search Performance

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[1]	Yes[1,2]	No	No	Yes[1]	Yes[1,4]
Optimal cost?	Yes[3]	Yes	No	No	Yes[3]	Yes[3,4]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$

Figure 3.15 Evaluation of search algorithms. b is the branching factor; m is the maximum depth of the search tree; d is the depth of the shallowest solution, or is m when there is no solution; ℓ is the depth limit. Superscript caveats are as follows: [1] complete if b is finite, and the state space either has a solution or is finite. [2] complete if all action costs are $\geq \epsilon > 0$; [3] cost-optimal if action costs are all identical; [4] if both directions are breadth-first or uniform-cost.

Adapted from: Russell & Norvig, AI: A Modern Approach

Coding Example

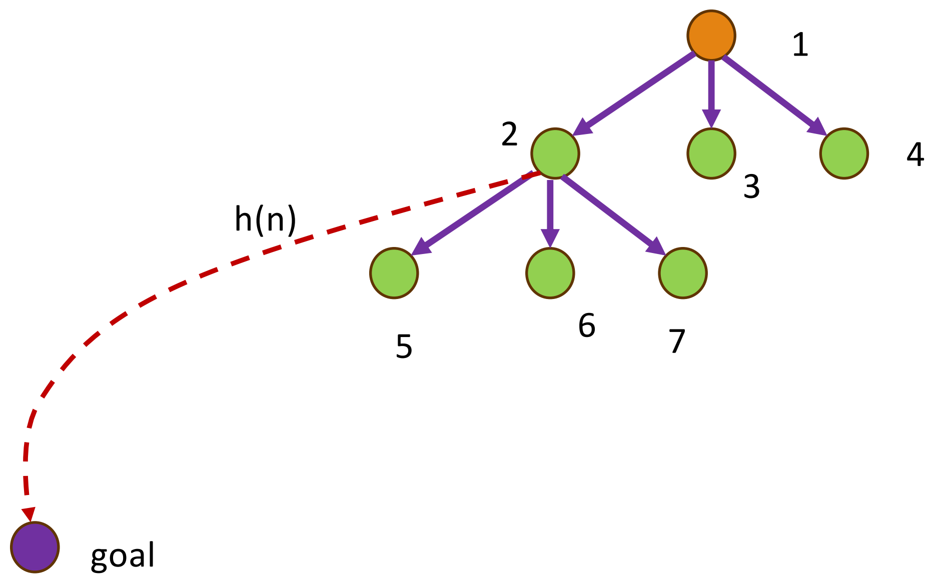
- N-Queens – code notebook
  - https://github.com/biplav-s/course-ai-tai-f23/blob/main/sample-code/Class6-To-Class10-search.md

Informed Search – Greedy best-first

Uses domain/problem specific hints to guide search

f(n) = h(n)

- f: estimated cost of best path via n to goal
- h: estimated cost to goal from n   // h is also called heuristic function

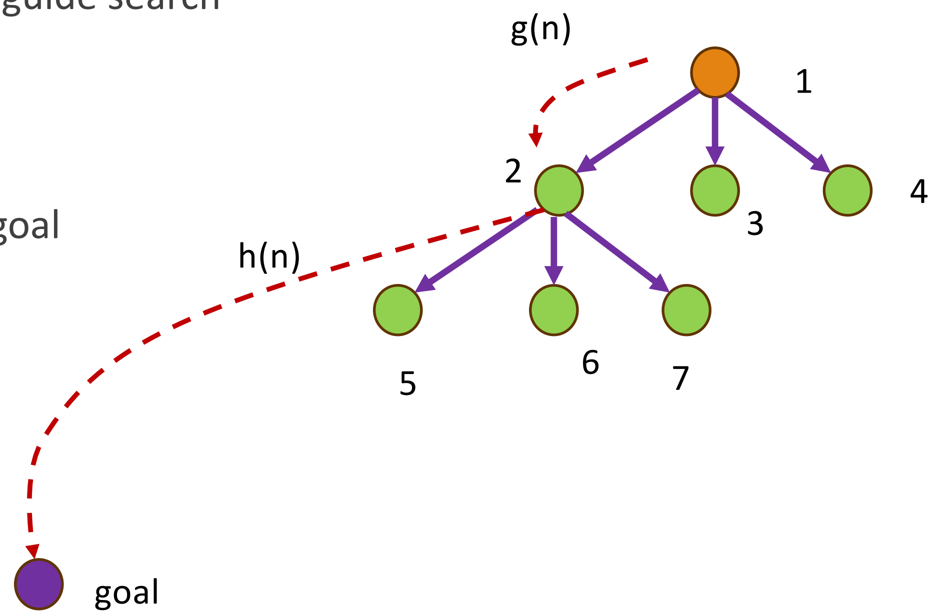


Informed Search – A* search

Uses domain/problem specific hints to guide search

f(n) = g(n) + h(n)

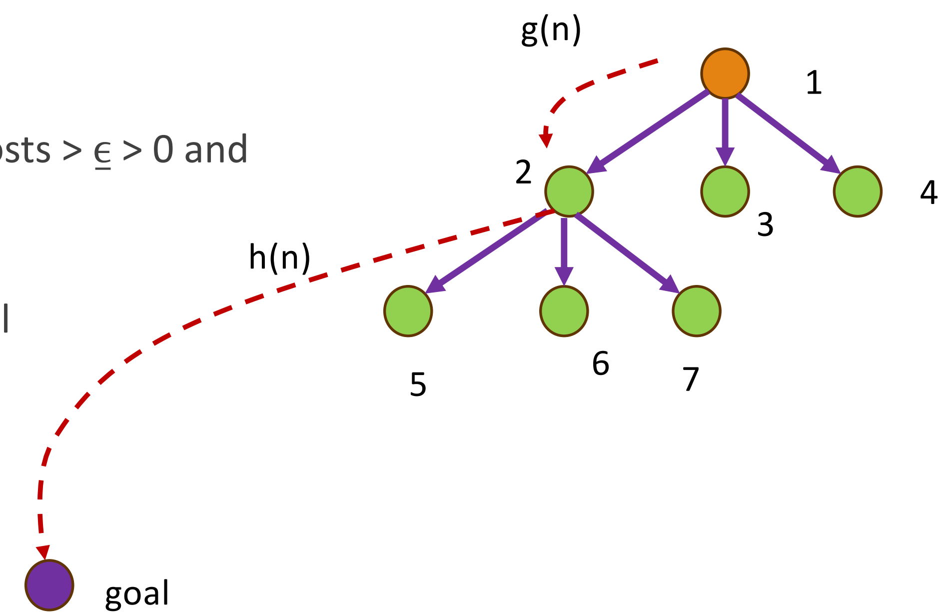
- f: estimated cost of best path via n to goal
- g: cost of best path to n
- h: estimated cost to goal from n



Properties of A* search

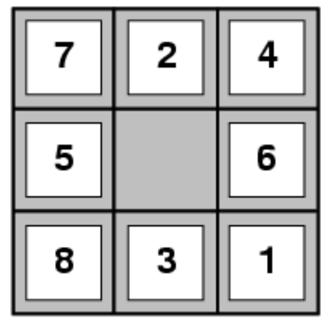
f(n) = g(n) + h(n)

- **A* is complete**, assuming actions costs > $\underline{\epsilon}$ > 0 and state space has solution or is finite
- **h(n) is admissible**, i.e., never overestimates true cost to reach goal

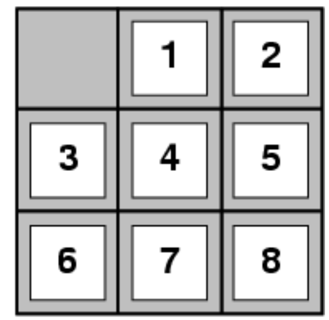


Finding Heuristics Function

- h1: number of misplaced tiles (excluding blank)
  - H1(start) = 8
- h2: sum of the distance of tiles from goal (excluding blank)
  - h2(start) = (3 + 1 + 2) + (2 + 3) + (2 + 2 + 3)
    = 18
- True cost: 26



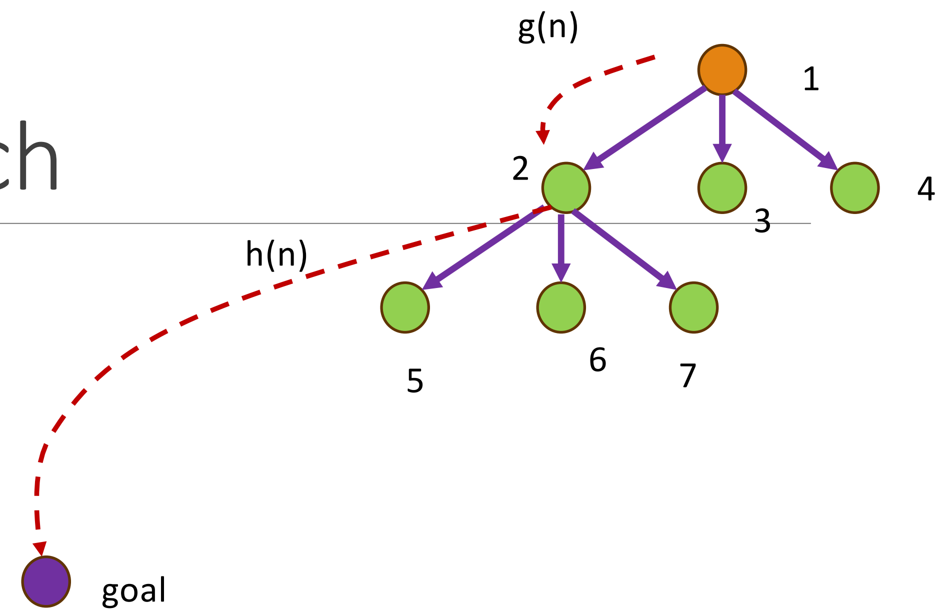
Start State



Goal State

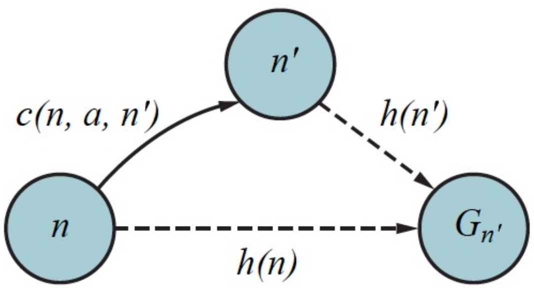
Adapted from:
Russell & Norvig, AI: A Modern Approach

Properties of A* search



f(n) = g(n) + h(n)

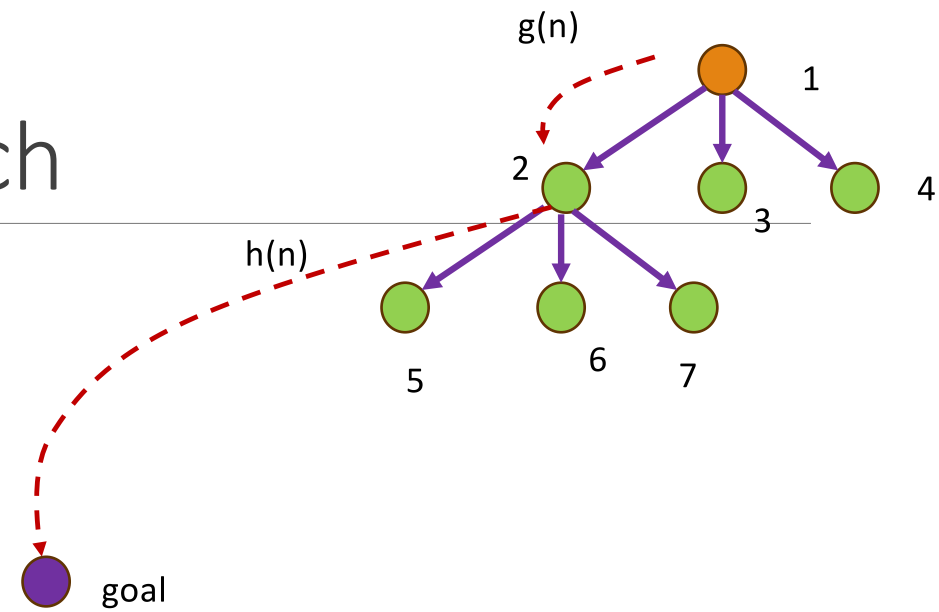
- A **heuristic is consistent** if
  h(n) <= c(n, a, n') + h(n')



Question: with a 'random' heuristic function be consistent?

Adapted from:
Russell & Norvig, AI: A Modern Approach

Properties of A* search



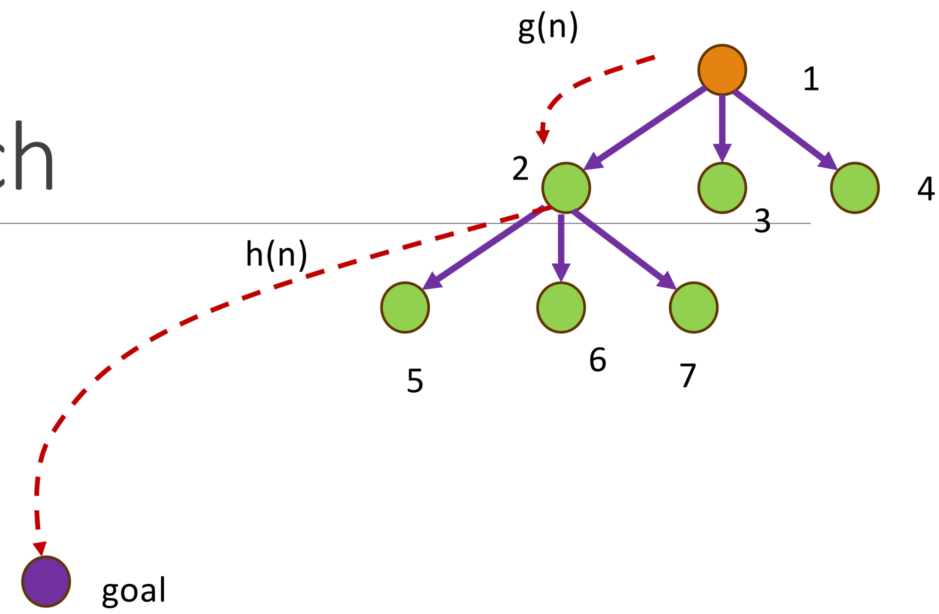
f(n) = g(n) + h(n)

- A* with consistent heuristic is **<u>optimally efficient</u>**

- A*
  - Any algo using search path and same heuristics as A* will at least expand these nodes
  - Prunes (removes) search nodes that are not necessary for finding optimal solution

Adapted from:
Russell & Norvig, AI: A Modern Approach

Type: Satisficing Search

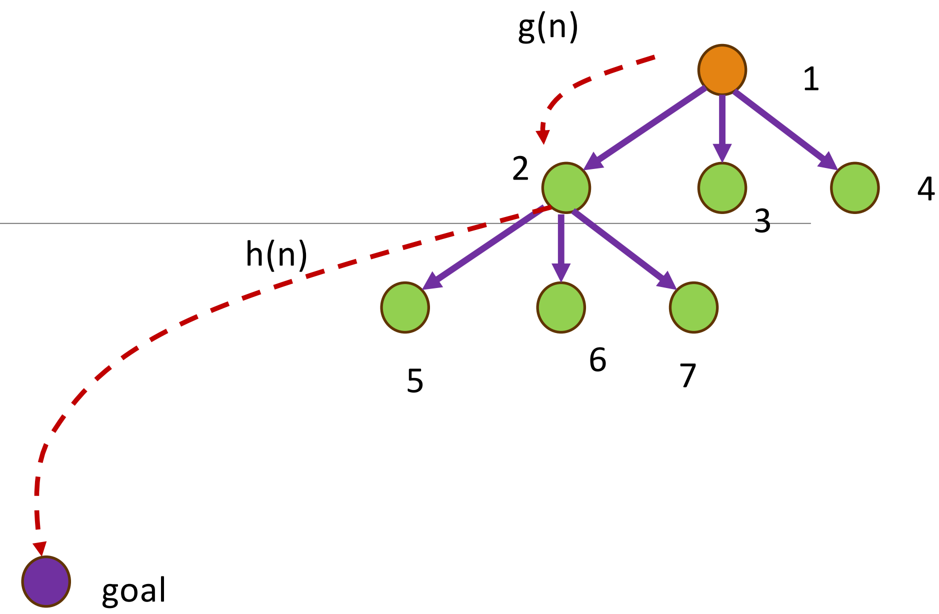


f(n) = g(n) + W * h(n)

- If heuristic is inadmissible, A* may find just any solution

Adapted from:
Russell & Norvig, AI: A Modern Approach

Type: Beam Search

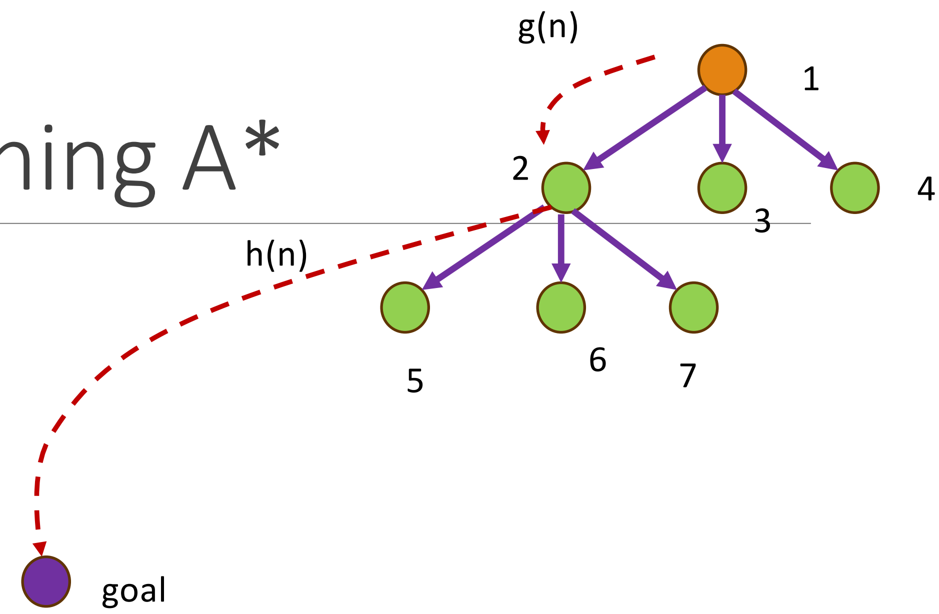


f(n) = g(n) + h(n)

- Keep only k (*a parameter*) nodes with the best f-score in frontier
- Incomplete and sub-optimal, but space efficient

Adapted from:
Russell & Norvig, AI: A Modern Approach

Type: Iterative Deepening A*



f(n) = g(n) + h(n)

- Similar to Iterative Deepening Depth search, but for f-score. Optimizes memory usage.
- In each iteration, search until find a node with f-score exceeding threshold; use the node's f-score as the new threshold
- Iterative search takes more time than plain A*. (Why?)

Adapted from:
Russell & Norvig, AI: A Modern Approach

Informed Search Types

A* search	f(n) = g(n) + h(n)	(W = 1)
Uniform-cost search	f(n) = g(n)	(W = 0)
Greedy best-first search	f(n) = h(n)	(W = 1)
Weighted A* search	f(n) = g(n) + W * h(n)	(1 < W < infinite)

Notes:
Uniform-cost => uninformed
Weighted A* => satisficing

Impact of Heuristics Function

	Search Cost (nodes generated)			Effective Branching Factor		
d	BFS	A*(h_1)	A*(h_2)	BFS	A*(h_1)	A*(h_2)
6	128	24	19	2.01	1.42	1.34
8	368	48	31	1.91	1.40	1.30
10	1033	116	48	1.85	1.43	1.27
12	2672	279	84	1.80	1.45	1.28
14	6783	678	174	1.77	1.47	1.31
16	17270	1683	364	1.74	1.48	1.32
18	41558	4102	751	1.72	1.49	1.34
20	91493	9905	1318	1.69	1.50	1.34
22	175921	22955	2548	1.66	1.50	1.34
24	290082	53039	5733	1.62	1.50	1.36
26	395355	110372	10080	1.58	1.50	1.35
28	463234	202565	22055	1.53	1.49	1.36

Figure 3.26 Comparison of the search costs and effective branching factors for 8-puzzle problems using breadth-first search, A* with h_1 (misplaced tiles), and A* with h_2 (Manhattan distance). Data are averaged over 100 puzzles for each solution length d from 6 to 28.

Reduces effective branching factor!

Adapted from:
Russell & Norvig, AI: A Modern Approach

Choosing From a Choice of (Admissible) Heuristics

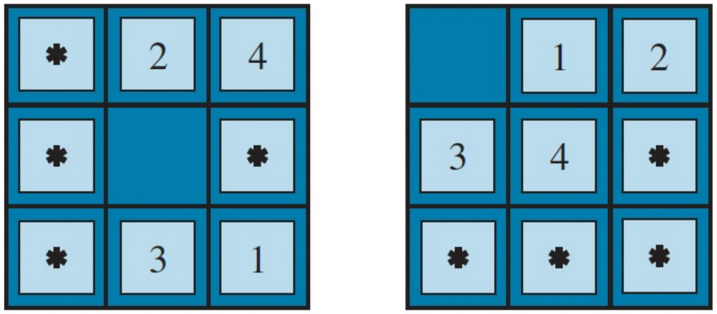
- Choose dominating heuristics
  - For n, h2(n) >= h1(n)

- If not dominating, choose maximum
  - $h(n) = \max \{h1(n), h2(n), \ldots, h_k(n)\}$

Creating Heuristics Automatically

- From relaxed problems
  - Formulate a relaxed problem
  - Solve relaxed problem
  - Use solution length as heuristics for original problem (Relaxed problem heuristics)
- From sub-problems
  - Formulate a sub-problem
  - Solve relaxed sub-problem
  - Store solution of sub-problem
  - Compute admissible heuristic h_DB for each node by looking up sub-problem and its solution cost

    (Pattern databases)
- Learn heuristics
  - From data: past solutions, relaxed problems, …
  - **Predict heuristic value**



Start State Goal State

Adapted from:
Russell & Norvig, AI: A Modern Approach

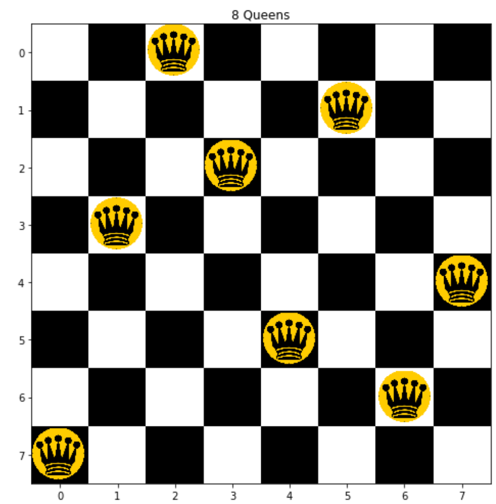
Coding Example

- 8-Puzzle – code notebook
  - https://github.com/biplav-s/course-ai-tai-f23/blob/main/sample-code/Class6-To-Class10-search.md

- From AIMA book
  - https://github.com/aimacode/aima-python/blob/master/search.ipynb
  - See 8-tile example

Discussion: Relaxed Problems

- For N-Queens
- Pancake problem
- Many more
https://www.movingai.com/SAS/index.html



Adapted from:
Russell & Norvig, AI: A Modern Approach

Discussion: Rubic’s Cube

- Search and deep-learning
  - Demo video: Solving with search and distance-based heuristics
    https://youtu.be/YQZ2sj-x5js
  - Live demo: Solving with A*search and deep learning-based heuristics (DeepCube-A)
    https://deepcube.igb.uci.edu/

Informed Search – A* search

- Best-first
- A*
- Weighted A*
- Beam search [Incomplete]
- Iterative-deepening A* [Incomplete]

A* search	f(n) = g(n) + h(n)	(W = 1)
Uniform-cost search	f(n) = g(n)	(W = 0)
Greedy best-first search	f(n) = h(n)	(W = 1)
Weighted A* search	f(n) = g(n) + W * h(n)	(1 < W < infinite)

Lecture 16: Summary

- We talked about
  - Informed Search
  - Heuristics and Properties
  - Designing Heuristics

Lecture 14:
Search, Search - Uninformed

Lecture 14: Outline

We will discuss

- Problems and representation: vacuum, sliding tile, N-queens
- Search – uninformed methods
- Analyzing search performance
  - Breadth-first search
  - Uniform-cost search
  - Depth-first search
  - Depth-limited search
  - Iterative deepening search
  - Bidirectional search

Lecture 17: Local Search

Lecture 17: Outline

We will discuss

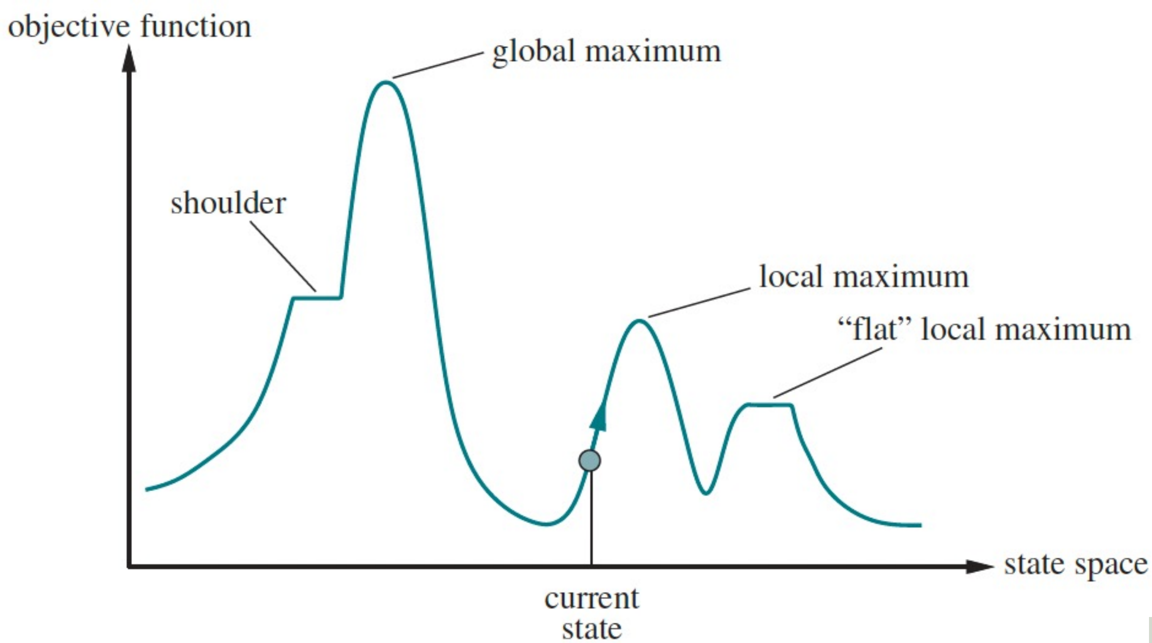
- Searching in large spaces
  - Hill climbing
  - Simulated Annealing
  - Genetic programming

Local Search

- Systematic search
  - Path matters [Store search trajectory]
- Non-systematic search
  - Solution matters, not path

- Settings
  - States: Discrete, continuous
  - Non-deterministic actions
  - Partial observability

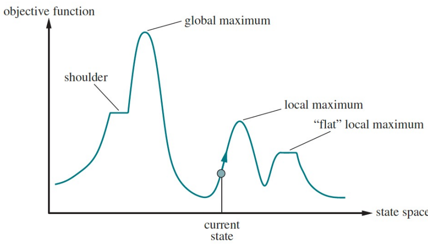
State Space Landscape



- Setup: find maxima

Adapted from:
Russell & Norvig, AI: A Modern Approach

Hill Climbing /Greedy Local Search



function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum
 current ← *problem*.INITIAL
 while *true* **do**
  neighbor ← a highest-valued successor state of *current*
  if VALUE(*neighbor*) ≤ VALUE(*current*) **then return** *current*
  current ← *neighbor*

At each step, replace the current node with the **best** neighbor.

Adapted from:
Russell & Norvig, AI: A Modern Approach

Hill Climbing Illustration

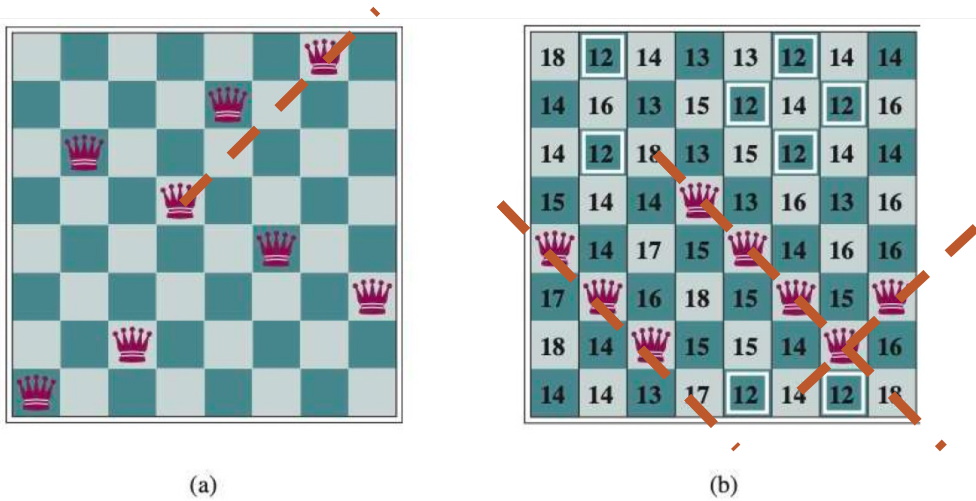


Figure 4.3 (a) The 8-queens problem: place 8 queens on a chess board so that no queen attacks another. (A queen attacks any piece in the same row, column, or diagonal.) This position is almost a solution, except for the two queens in the fourth and seventh columns that attack each other along the diagonal. (b) An 8-queens state with heuristic cost estimate $h = 17$. The board shows the value of h for each possible successor obtained by moving a queen within its column. There are 8 moves that are tied for best, with $h = 12$. The hill-climbing algorithm will pick one of these.

State representation:

- Complete state formulation

Next Action:

- Any queen in the same column
  (8 x 7 = 56 children)

State space: 8^8 = 17 million (appx)

Steepest ascent:

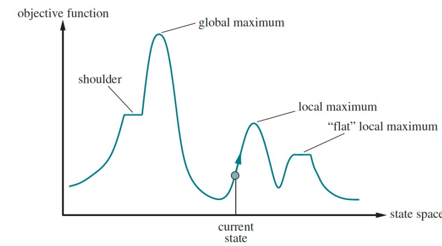
* Gets stuck 86% times in 3 steps
* Solves 14% times in 4 steps

Adapted from:
Russell & Norvig, AI: A Modern Approach

Hill Climbing Variations

- **Stochastic hill climbing**: chooses an uphill node with prob. depending on steepness of increase
- **First-choice hill climbing**: choose first that is uphill
- **Random-restart hill climbing**: restart after a few tries
  - If p is chance of success. restarts needed = 1/p
  - For 8-queens, p=.14
    - Restart needed = 7 (6 failure, 1 success)
    - Total steps for finding a solution = 4 +((1-p) / p) * 3 = 22 steps

Simulated Annealing



function SIMULATED-ANNEALING($problem$, $schedule$) **returns** a solution state
  $current \leftarrow problem$.INITIAL
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** $current$
    $next \leftarrow$ a randomly selected successor of $current$
    $\Delta E \leftarrow$ VALUE($current$) – VALUE($next$)
    if $\Delta E > 0$ **then** $current \leftarrow next$
    else $current \leftarrow next$ only with probability $e^{-\Delta E/T}$

- Setup: find minima
- T: temperature
- A bad successor is chosen will prob. that decreases with temperature
- Schedule: cooling schedule

Adapted from:
Russell & Norvig, AI: A Modern Approach

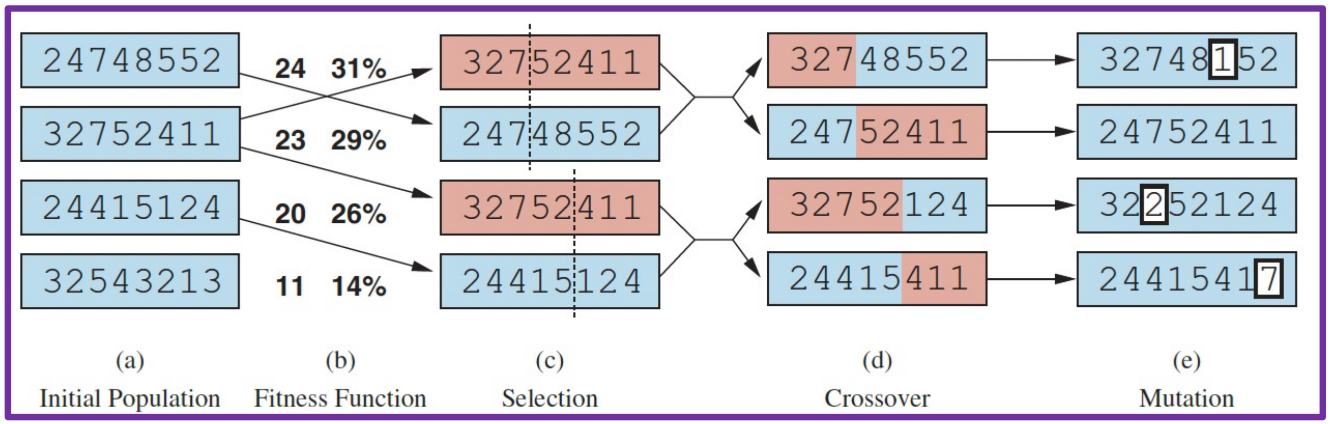
Related Algorithms

- **Local beam search**: keeps track of **k** states rather than 1
  - Generate k randomly generated states
  - Repeat
    - Generate all successors of k states generated
    - If one is a goal, done
    - Select k best successors
- Stochastic beam search
  - Chooses k successors with probability proportional to the successors value

Evolutionary Algorithms (EAs)

Basic idea

- A population of individuals (states)
- Fittest (highest value) produce offsprings (successor states) - recombination
  - Cross-over
  - mutation



Digit strings representing 8-queens states. The initial population in (a) is ranked by a fitness function in (b) resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

Fitness function: non-attacking pairs of queens

Adapted from:
Russell & Norvig, AI: A Modern Approach

Comparing EA with Local Search

- Idea of cross-over
  - Useful if traits of parents are useful in children
- Idea of mutation
  - Random changes can help escape local minima
- Selection of parameters (e.g., generations) affects performance
- # Parents
  - =1 : stochastic beam search
  - =2 : similar to nature
  - > 2 : not common in nature, but possible to simulate

```
function GENETIC-ALGORITHM(population, fitness) returns an individual
  repeat
     weights ← WEIGHTED-BY(population, fitness)
     population2 ← empty list
     for i = 1 to SIZE(population) do
        parent1, parent2 ← WEIGHTED-RANDOM-CHOICES(population, weights, 2)
        child ← REPRODUCE(parent1, parent2)
        if (small random probability) then child ← MUTATE(child)
        add child to population2
     population ← population2
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to fitness

function REPRODUCE(parent1, parent2) returns an individual
  n ← LENGTH(parent1)
  c ← random number from 1 to n
  return APPEND(SUBSTRING(parent1, 1, c), SUBSTRING(parent2, c + 1, n))
```

Figure 4.7 A genetic algorithm. Within the function, *population* is an ordered list of individuals, *weights* is a list of corresponding fitness values for each individual, and *fitness* is a function to compute these values.

Adapted from:
Russell & Norvig, AI: A Modern Approach

Local Search With Non-Deterministic Actions

- Systematic search
  - Path matters [Store search trajectory]
- Non-systematic search
  - Solution matters, not path
- Settings
  - States: Discrete, continuous
  - **Non-deterministic actions***
  - Partial observability*

Erratic Vacuum World

- When applied to a dirty square, the robot cleans that room and sometimes the adjacent room
- When applied to a clean square, the robot throws dirt in the room

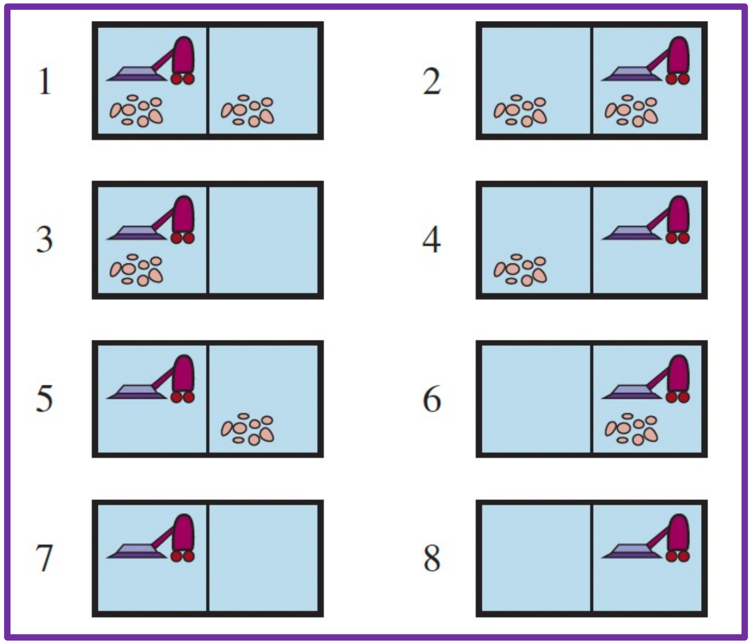
Adapted from:
Russell & Norvig, AI: A Modern Approach

* Solutions are not nodes but conditional plans/ strategies.

Local Search With Non-Deterministic Actions

Erratic Vacuum World

- When applied to a dirty square, the robot cleans that room and sometimes the adjacent room
- When applied to a clean square, the robot throws dirt in the room

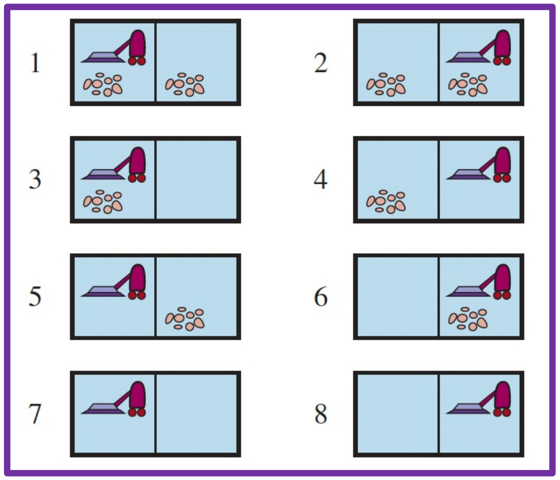


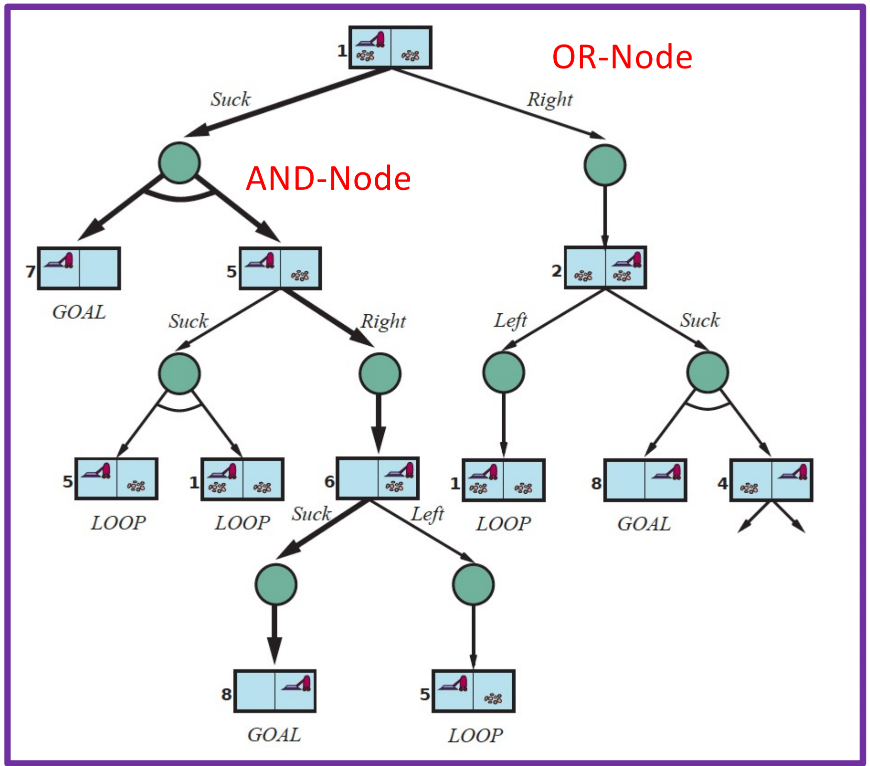
Adapted from:
Russell & Norvig, AI: A Modern Approach

Local Search With Non-Deterministic Actions

Erratic Vacuum World

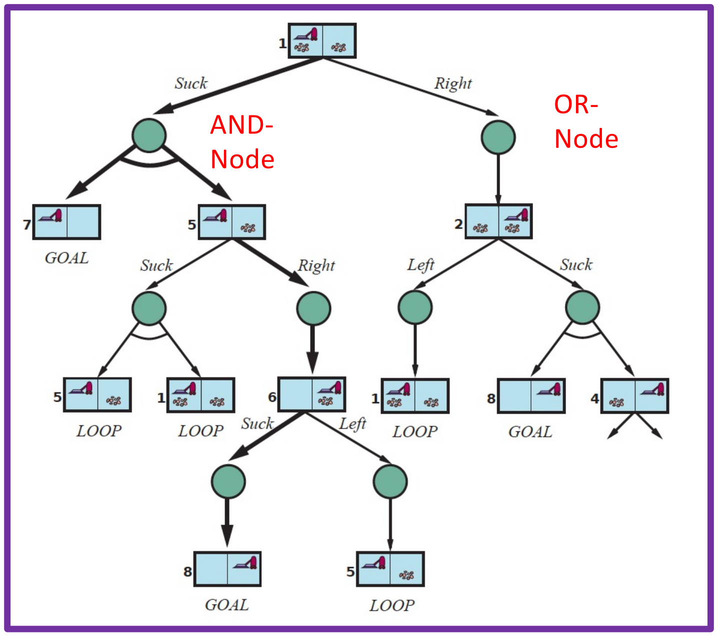
- When applied to a dirty square, the robot cleans that room and sometimes the adjacent room
- When applied to a clean square, the robot throws dirt in the room





Adapted from:
Russell & Norvig, AI: A Modern Approach

Local Search With Non-Deterministic Actions




```
function AND-OR-SEARCH(problem) returns a conditional plan, or failure
   return OR-SEARCH(problem, problem.INITIAL, [])

function OR-SEARCH(problem, state, path) returns a conditional plan, or failure
   if problem.IS-GOAL(state) then return the empty plan
   if IS-CYCLE(path) then return failure
   for each action in problem.ACTIONS(state) do
      plan ← AND-SEARCH(problem, RESULTS(state, action), [state] + path])
      if plan ≠ failure then return [action] + plan]
   return failure

function AND-SEARCH(problem, states, path) returns a conditional plan, or failure
   for each s_i in states do
      plan_i ← OR-SEARCH(problem, s_i, path)
      if plan_i = failure then return failure
   return [if s_1 then plan_1 else if s_2 then plan_2 else ... if s_{n-1} then plan_{n-1} else plan_n]
```

Adapted from:
Russell & Norvig, AI: A Modern Approach

Coding Example

- 8-Puzzle – code notebook
  - https://github.com/biplav-s/course-ai-tai-f23/blob/main/sample-code/Class6-To-Class10-search.md

Lecture 17: Summary

- We talked about
  - Hill climbing
  - Simulated Annealing
  - Genetic programming
  - Search in complex environments

Week 9: Concluding Comments

We talked about

- Informed search
- Local search

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Projects B: Sep 30 – Nov 20 (7 weeks; 400 points)

- End date: **Thursday, Nov 20**
  - Remember to update spreadsheet on data/ time when finished **(Column I)**

- Choices
  - Given by instructor
  - Defined by student using project-b teamplate; reviewed and approved by instructor

Upcoming Evaluation Milestones

- Projects B: Sep 30 – Nov 20
- Quiz 2: Oct 7
- Quiz 3: Nov 11
- Paper presentation (grad students only) : Nov 18
- Finals: Dec 11

About Week 10 – Lectures 18, 19

Week 10 – Lectures 18, 19

- Lecture 18: Adversarial games and search
- Lecture 19: Constraints & optimization

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2: Data: Formats, Representation, ML Basics
- Week 3: Machine Learning – Supervised (Classification)
- Week 4: Machine Learning - Unsupervised (Clustering) –
- Topic 5: Learning neural network, deep learning, Adversarial attacks
- Week 6: Large Language Models – Representation and Usage issues
- Weeks 7-8: Search, Heuristics - Decision Making
- Week 9: Constraints, Optimization – Decision Making
- Topic 10: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 11-12: Planning, Reinforcement Learning – Sequential decision making
- Week 13: Trustworthy Decision Making: Explanation, AI testing
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Note: exact schedule changes slightly to accommodate for exams and holidays.