CSCE 580: Introduction to AI

Lecture 11: Constraints and Optimization

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

24TH SEP 2024

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

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

Organization of Lecture 11

- Introduction Segment
  - Recap of Lecture 10
- Main Segment
  - Constraint Satisfaction Problem
  - Optimization Problems
- Concluding Segment
  - Course Project Discussion
  - About Next Lecture – Lecture 12
  - Ask me anything

Introduction Section

Recap of Lecture 10

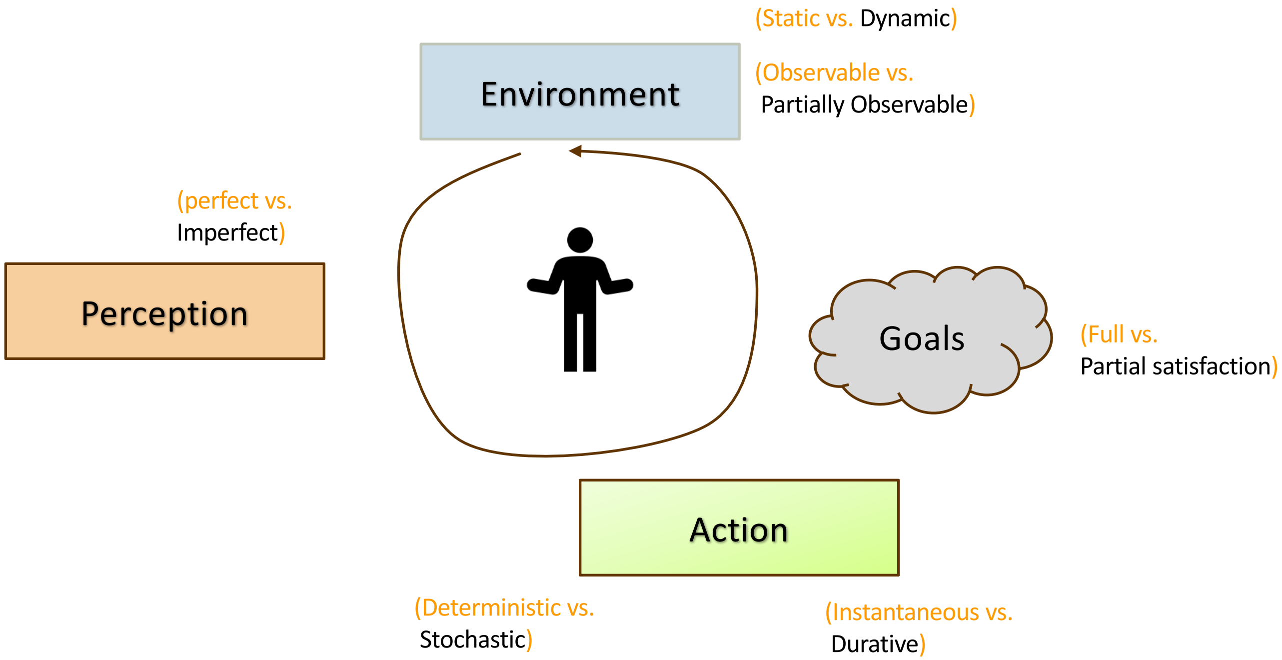
- Games and Search
- Minimax
- Alphabeta
- Monte Carlo Tree Search

- Overall - search topics covered
  - Formulating search problems
  - Uninformed search
  - Informed search
  - Local search
  - Game search

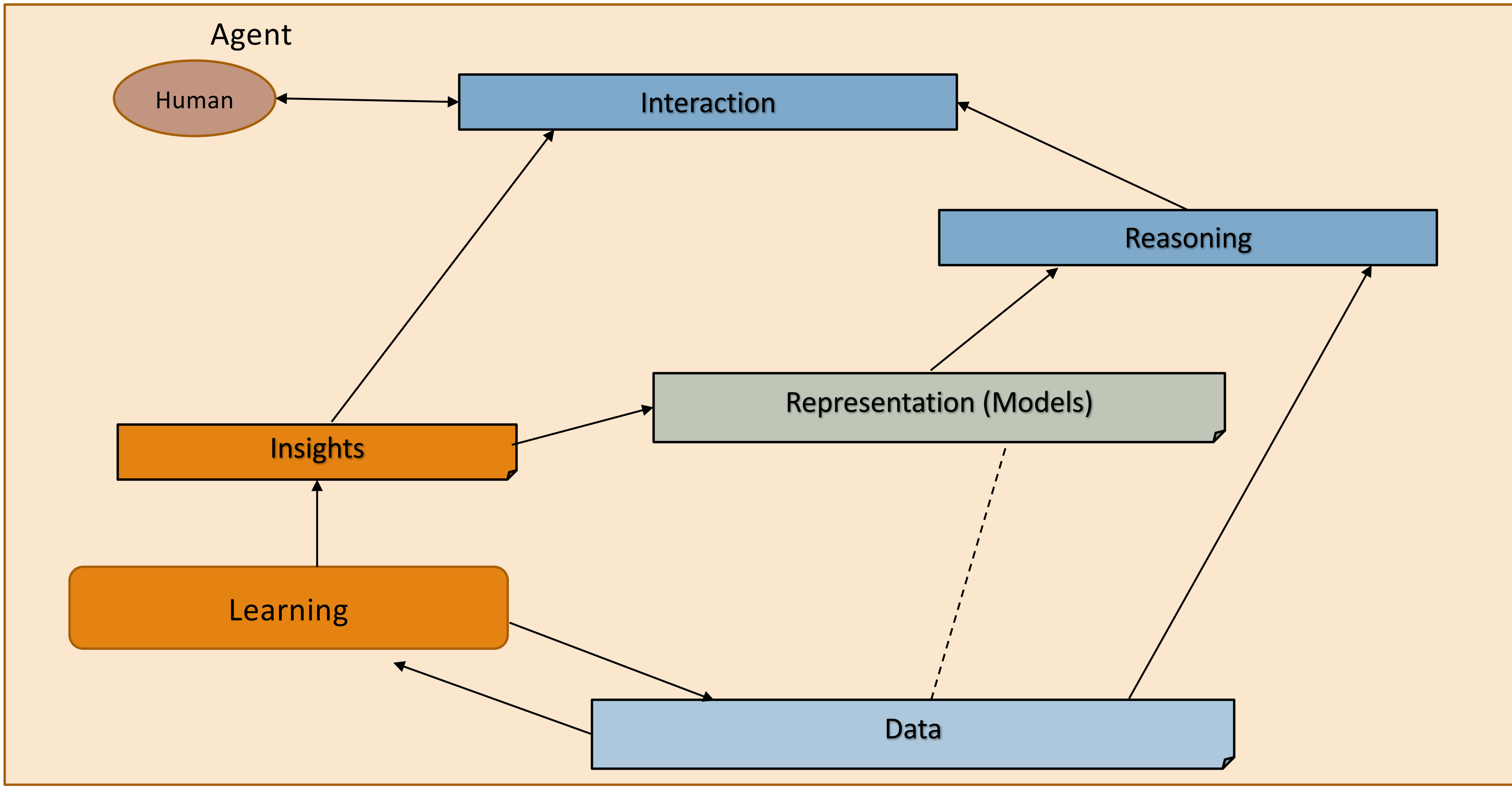
Quiz 1 - Grading

- Follow instructions
  - Not writing name: (-1)
  - Not answering all questions lead to loss of marks
  - Not providing access to code for Q3 lead to loss of marks
- Grading was generous
- Some students were exceptional

Intelligent Agent Model



Relationship Between Main AI Topics



Where We Are in the Course

CSCE 580/ 581 – In This Course

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

Main Section

Constraint Satisfaction Problems (CSPs)

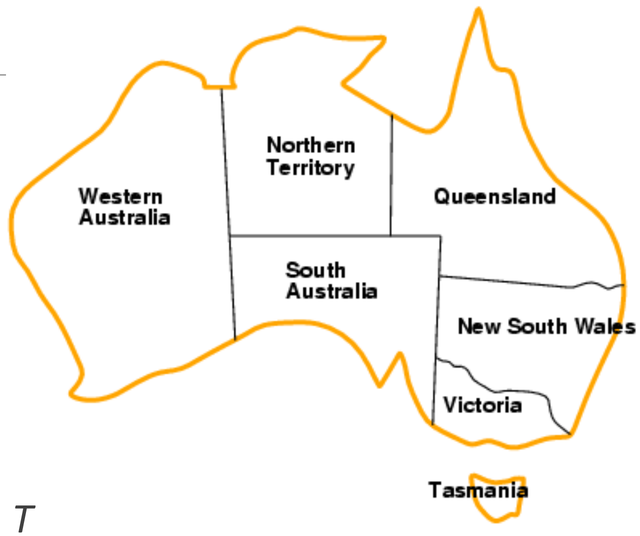
- X - A set of variables $\{X_1, \ldots, X_n\}$
- D - A set of domains $\{D_1, \ldots, D_n\}$, for each variable
- C – set of constraints specifying allowed combinations of values for variables

Example

- $X_1 = \{1,2,3\}$, $X_2 = \{1,2,3\}$
  - Here, $D_1 = D_2 = \{1,2,3\}$
- $C = \langle (X_1, X_2), X_1 > X_2 \rangle$

Solutions = Assignments to $(X_1, X_2) = \{(3,1), (3,2), (2,1)\}$

Example: Map-Coloring



Variables *WA, NT, Q, NSW, V, SA, T*

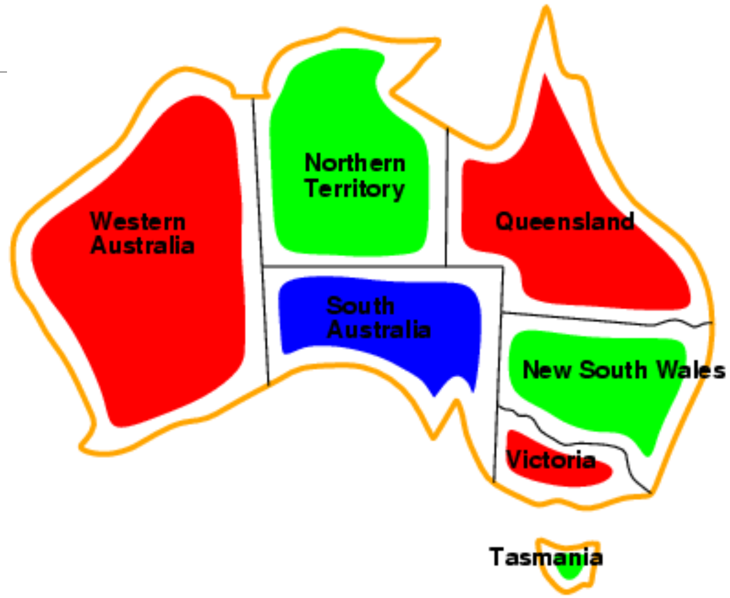
Domains D_i = {red, green, blue}

Constraints: adjacent regions must have different colors

e.g., WA ≠ NT, or (WA,NT) in {(red,green),(red,blue),(green,red), (green,blue),(blue,red),(blue,green)}

Adapted from:
Tuomas Sandholm's CSP Lecture
Russell & Norvig, AI: A Modern Approach

Example: Map-Coloring



Solutions are complete and consistent assignments

e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue,T = green

Adapted from:
Tuomas Sandholm's CSP Lecture
Russell & Norvig, AI: A Modern Approach

Varieties of CSPs

Discrete variables

- finite domains:
  - n variables, domain size d → $O(d^n)$ complete assignments
  - e.g., Boolean CSPs, includes Boolean satisfiability (NP-complete)
- infinite domains:
  - integers, strings, etc.
  - e.g., job scheduling, variables are start/end days for each job
  - need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$

Continuous variables

- e.g., start/end times for Hubble Space Telescope observations
- linear constraints solvable in polynomial time by LP

Adapted from:
Tuomas Sandholm's CSP Lecture
Russell & Norvig, AI: A Modern Approach

Varieties of constraints

Unary constraints involve a single variable,

- e.g., SA ≠ green

Binary constraints involve pairs of variables,

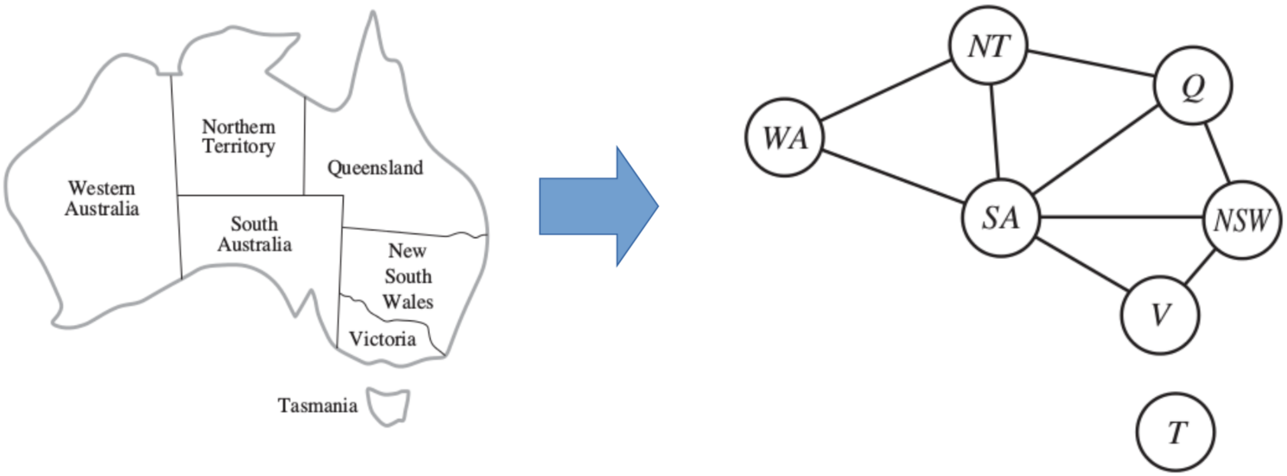
- e.g., SA ≠ WA

Higher-order constraints involve 3 or more variables,

- e.g., cryptarithmetic column constraints

Adapted from:
Tuomas Sandholm's CSP Lecture
Russell & Norvig, AI: A Modern Approach

The constraint graph



Binary CSP: each constraint relates at most two variables

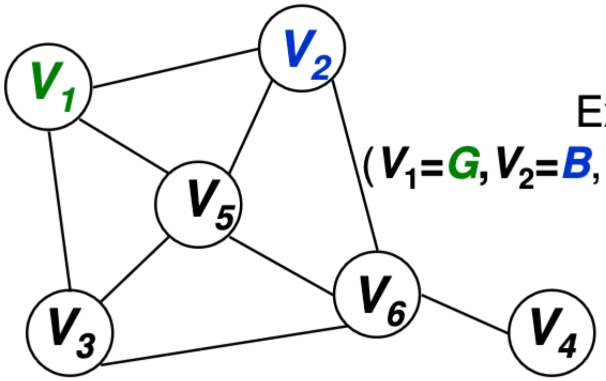
Constraint graph: nodes are variables, arcs show constraints

General-purpose CSP algorithms use the graph structure to speed up search

E.g., Tasmania is an independent subproblem!

Adapted from:
- https://www.khoury.northeastern.edu/home/camato/5100/csp.pdf
- https://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020107CSP.pdf

Search Space



Example state:
(V_1=**G**, V_2=**B**, V_3=?, V_4=?, V_5=?, V_6=?)

- *State*: assignment to k variables with k+1,..,N unassigned
- *Successor*: The successor of a state is obtained by assigning a value to variable k+1, keeping the others unchanged
- *Start state*: (V_1=?, V_2=?, V_3=?, V_4=?, V_5=?, V_6=?)
- *Goal state*: All variables assigned with constraints satisfied
- No concept of cost on transition → We just want to find a solution, we don't worry how we get there

Adapted from:
- Tuomas Sandholm's CSP Lecture
- https://www.khoury.northeastern.edu/home/camato/5100/csp.pdf
- https://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020107CSP.pdf

Example: Cryptarithmetic

- Variables

  D, E, M, N, O, R, S, Y
- Domains

  {0, 1, 2, 3 ,4, 5, 6, 7, 8, 9 }
- Constraints

  $M \neq 0$, $S \neq 0$ (unary constraints)

  $Y = D + E$ OR $Y = D + E - 10$.

  $D \neq E$, $D \neq M$, $D \neq N$, etc.

```
  S E N D
+ M O R E
---------
M O N E Y
```

Adapted from:
- Tuomas Sandholm's CSP Lecture
- https://www.khoury.northeastern.edu/home/camato/5100/csp.pdf
- https://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020107CSP.pdf

A harder CSP to represent: Cryptarithmetic

- **Variables:**

$F \ T \ U \ W \ R \ O \ X_1 \ X_2 \ X_3$

- **Domains:**

$\{0,1,2,3,4,5,6,7,8,9\}$

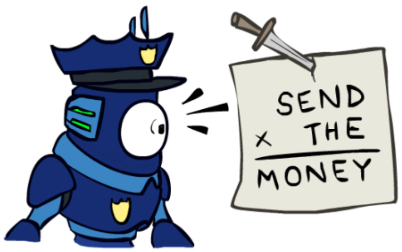
- **Constraints:**

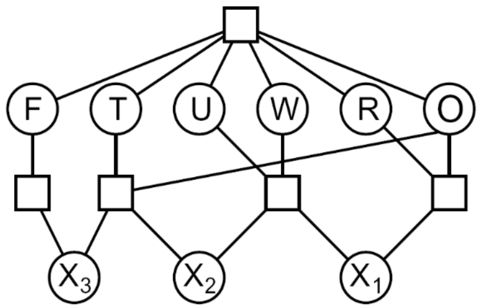
$\text{alldiff}(F,T,U,W,R,O)$

$O + O = R + 10 \cdot X_1$

$\cdots$

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$

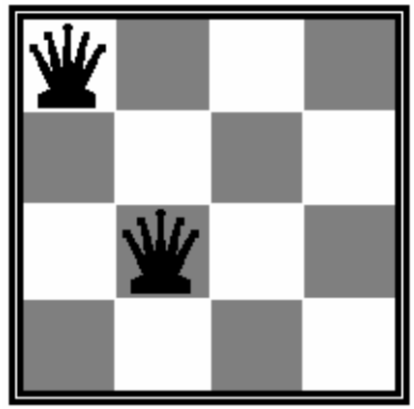




Adapted from:
- https://www.khoury.northeastern.edu/home/camato/5100/csp.pdf
- https://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020107CSP.pdf

Example: N-Queens

- Variables: Q_i
- Domains: $D_i = \{1, 2, 3, 4\}$
- Constraints
  - $Q_i \neq Q_j$ (cannot be in same row)
  - $|Q_i - Q_j| \neq |i - j|$ (or same diagonal)



$Q_1 = 1 \quad Q_2 = 3$

- Valid values for (Q_1, Q_2) are (1,3) (1,4) (2,4) (3,1) (4,1) (4,2)

Adapted from:
- Tuomas Sandholm's CSP Lecture
- https://www.khoury.northeastern.edu/home/camato/5100/csp.pdf
- https://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020107CSP.pdf

Constraint Propagation

- **Node Consistency**: a variable (node in CSP graph) is node—consistent of all the values in the variable's domain satisfy variable's unary constraints
- **Arc Consistency**: every variable in its domain satisfies binary constraints

function AC-3(csp) **returns** false if an inconsistency is found and true otherwise
  $queue \leftarrow$ a queue of arcs, initially all the arcs in csp

  while $queue$ is not empty **do**
    $(X_i,\ X_j) \leftarrow$ POP($queue$)
    if REVISE(csp, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** $false$
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add $(X_k,\ X_i)$ to $queue$
  return $true$

function REVISE(csp, X_i, X_j) **returns** true iff we revise the domain of X_i
  $revised \leftarrow false$
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      $revised \leftarrow true$
  return $revised$

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

Constraint Propagation

- **Path Consistency**: A two variables set $\{X_i , X_j\}$ is path-consistent with respect to a third variable X_m if, for every assignment $\{X_i = a_i, X_j = a_j\}$ consistent with constraints on $\{X_i , X_j\}$, there is an assignment to X_m which is consistent with $\{X_i , X_m\}$ and $\{X_m , X_j\}$

- **k-consistency**: A CSP is k-consistent if for any set of (k-1) variables and their consistent assignments, a consistent value can always be assigned for the k^{th} variable.

Coding Example

- CSP code notebook
  - https://github.com/aimacode/aima-python/blob/master/csp.ipynb/

Making Optimal Decisions

Optimal Decision

- What is it? There is no absolute answer. In AI, there is the concept of a **rational** agent.
- Acting rationally: acting such that one ca achieve one's goals given one's beliefs (and information)
  - But what are one's goals
  - Are the always of achievement?
- Some options
  - Perfect rationality: maximize expected utility at every time instant
    - Given the available information; can be computationally expensive
    - "Doing the right thing"
  - Bounded optimality: do as well as possible given computational resources
    - Expected utility as high as any other agent with similar resources
  - Calculative rationality: *eventually* returns what would have been the rational choice

What Is It?

- As a working principle
  - Bounded or Calculative Rationality
- In observable and deterministic scenarios
  - Maximize utility: (benefit – cost)
- In scenarios with uncertainty and/ or unobservable
  - Maximize ***expected*** utility: (benefit – cost)

Example Situation – Course Selection

- A person wants to pass an academic program in two majors: A and B
- There are three subjects: A, B and C, each with three levels (*1, *2, *3). There are thus 9 courses: A1, A2, A3, B1, B2, B3, C1, C2, C3
- To graduate, at least one course at beginner (*1) level is needed in major(s) of choice(s), and two courses at intermediate levels (*2) are needed
- **Optimality considerations** in the problem
  - Least courses, fastest time to graduate, class size, friends attending together, …
- Answer questions
  - Q1: How many <u>minimum</u> courses does the person have to take ?
  - Q2: Can a person graduate in 2 majors studying 3 courses only?
  - …

Algorithms for Optimality

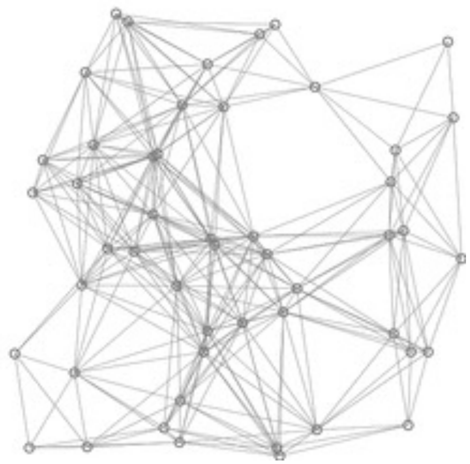
- Problem specific methods
  - Path finding
  - Linear programming
  - Constraint satisfaction and optimization

- General-purpose methods for optimality using search

Optimality: Example - Path Finding

Main steps

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the *unvisited set*.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances through the current node. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one.
4. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.
5. If the destination node has been marked visited or if the smallest tentative distance among the nodes in the *unvisited set* is infinity, then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.



A demo of Dijkstra's algorithm based on Euclidean distance

Djikstra's Algorithm with positive numbers or labels that are monotonically non-decreasing.

Source: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Exercise and Code

- Linear Programming Methods

  - Link - https://github.com/biplav-s/course-d2d-ai/blob/main/sample-code/l16-optimal/Optimization.ipynb

Lecture 10: Summary

- We talked about
  - Constraint Satisfaction Problem
  - Optimization Problems

Concluding Section

Course Project

Discussion: Projects

- New: two projects
  - Project 1: model assignment
  - Project 2: single problem/ llm based solving / fine-tuning/ presenting result

Project Discussion

1. Go to Google spreadsheet against your name
2. Enter model assignment name and link from (http://modelai.gettysburg.edu/)

1. Create a private Github repository called "CSCE58x-Fall2024-<studentname>-Repo". Share with Instructor (biplav-s) and TA (vishalpallagani)
2. Create Google folder called "CSCE58x-Fall2024-<studentname>-SharedInfo". Share with Instructor (prof.biplav@gmail.com) and TA (vishal.pallagani@gmail.com)
3. Create a Google doc in your Google repo called "Project Plan" and have the following by next class (Sep 5, 2024)

Timeline
1. Title:
2. Key idea: (2-3 lines)
3. Data need:
4. Methods:
5. Evaluation:
6. Milestones
    1. // Create your own
7. Oct 3, 2024

Reference: Project 1 Rubric (30% of Course)

Assume total for Project-1 as 100

- **Project results** – 60%
  - Working system ? – 30%
  - Evaluation with results superior to baseline? – 20%
  - Went through project tasks completely ? – 10%
- **Project effort**s – 40%
  - Project report – 20%
  - Project presentation (updates, final) – 20%
- **Bonus**
  - Challenge level of problem – 10%
  - Instructor discretion – 10%
- **Penalty**
  - Lack of timeliness as per your milestones policy (right) - up to 30%

Milestones and **Penalties**

- Project plan due by Sep 5, 2024 **[-10%]**
- Project deliverables due by Oct 3, 2024 **[-10%]**
- Project presentation on Oct 8, 2024 **[-10%]**

About Next Lecture – Lecture 12

Lecture 12: ML Basics

- ML Problem Settings
- Data preparation and feature engineering
- Solving classification problems

7	Sep 10 (Tu)	Search - Uninformed
8	Sep 12 (Th)	Search - Informed; Heuristics
9	Sep 17 (Tu)	Local search
10	Sep 19 (Th)	Adversarial games and search
11	Sep 24 (Tu)	Constraints & optimization
12	Sep 26 (Th)	Machine Learning - Basics
13	Oct 1 (Tu)	Machine Learning – Classification – Decision Trees, Random Forest
14	Oct 3 (Th)	Machine Learning – Classification – NBC, Gradient Boosting, ML-Text