

*CSCE 580: Introduction to AI*  
*CSCE 581: Trusted AI*

## Lecture 11: Constraints and Optimization

---

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

28<sup>TH</sup> SEP 2023

**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

---

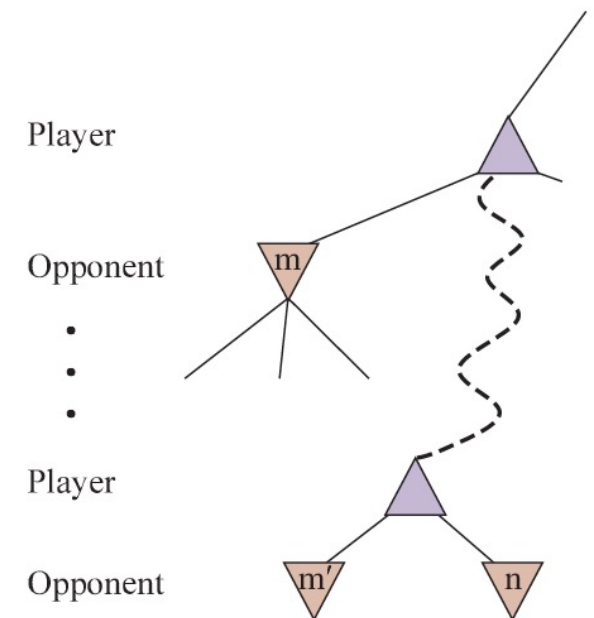
# Recap of Lecture 10

---

- Games and Search
  - Minimax
  - Alphabeta
  - Monte Carlo Tree Search
- Overall - search topics covered
  - Formulating search problems
  - Uninformed search
  - Informed search
  - Local search
  - Game search

# Errata - Impact of Alpha Beta

- MINIMAX( $O(b^m)$ )
- Cuts down on size of game tree searched
  - Reduction depends on order of nodes traversed
  - Assuming order of nodes is random, expect to prune after exploring half the branches
  - Best case:  $O(b^{m/2})$ , where  $b$  is branching factor and  $m$  is depth



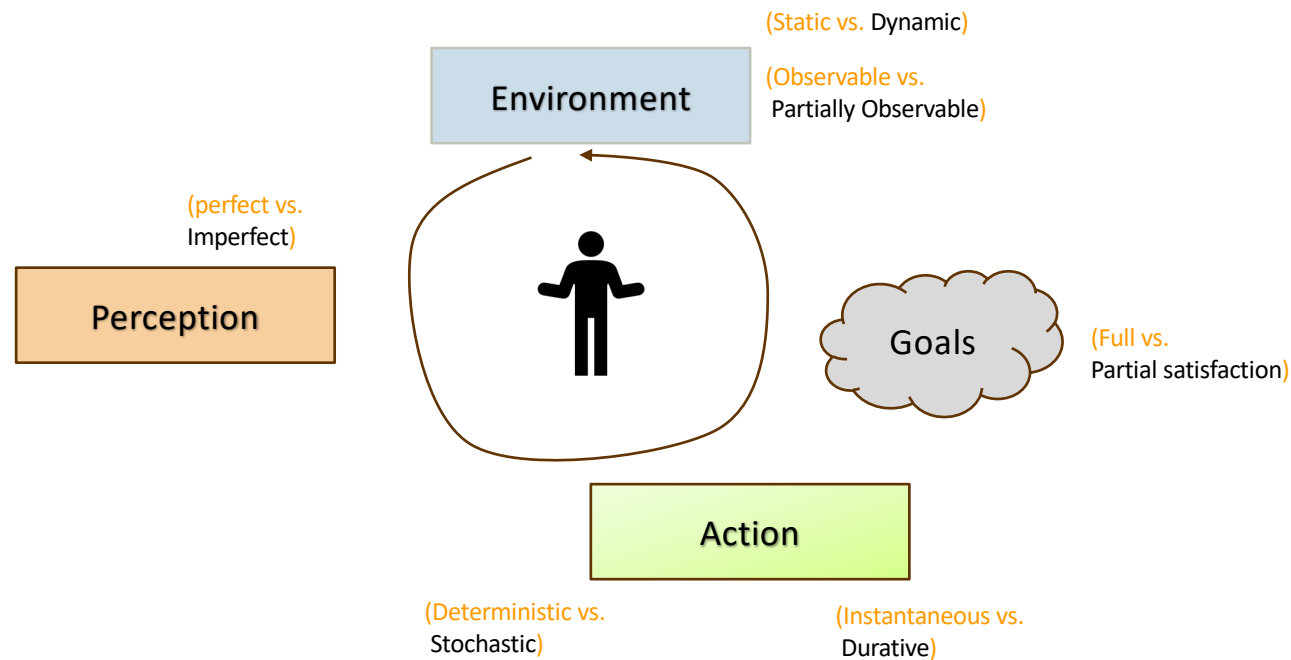
Adapted from:  
Russell & Norvig, AI: A Modern Approach

# Quiz 1 - Grading

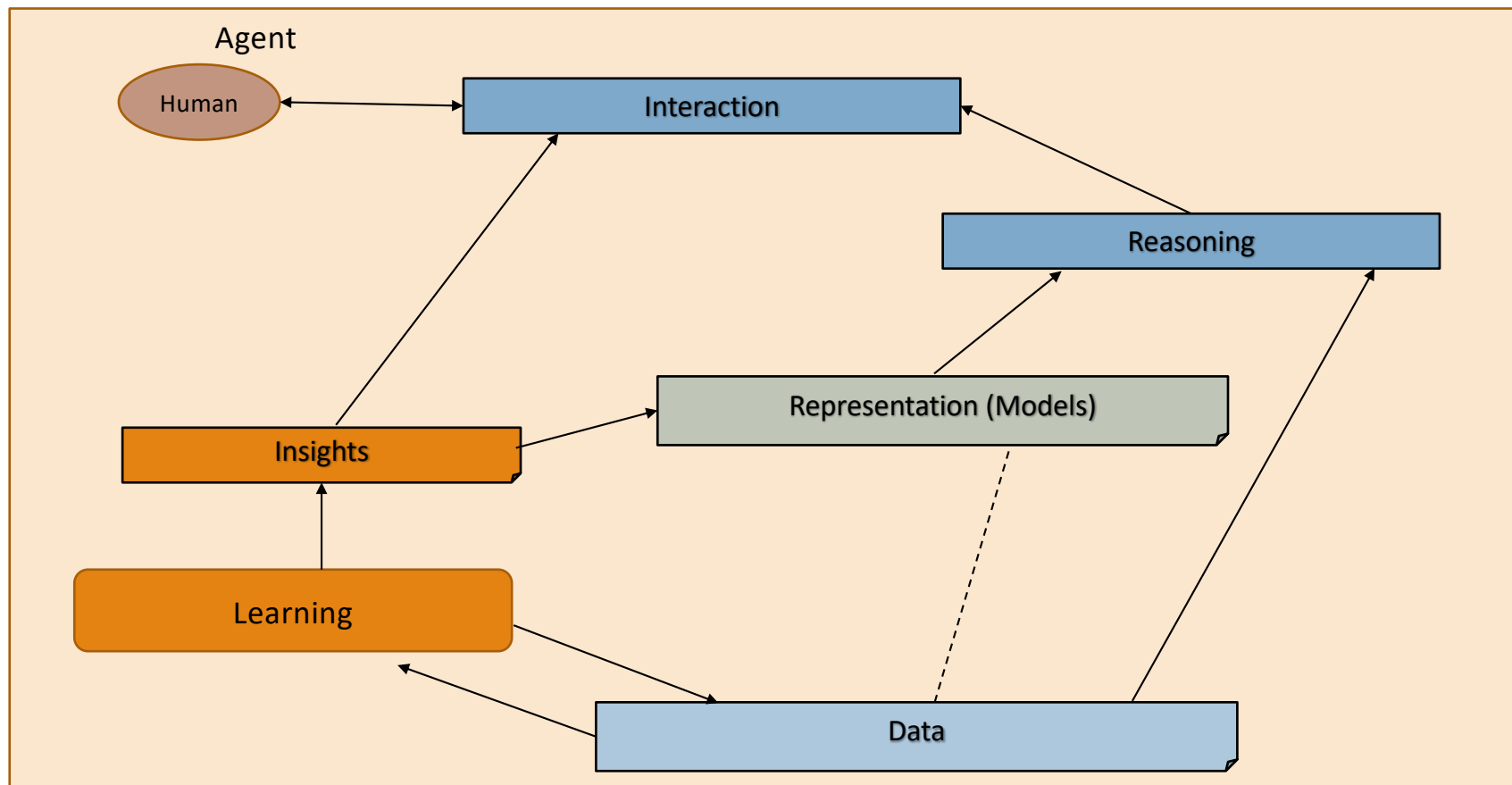
---

- 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**  $\{X_1, \dots, X_n\}$
- $D$  - A set of **domains**  $\{D_1, \dots, D_n\}$ , 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**  $WA, NT, Q, NSW, V, SA, T$

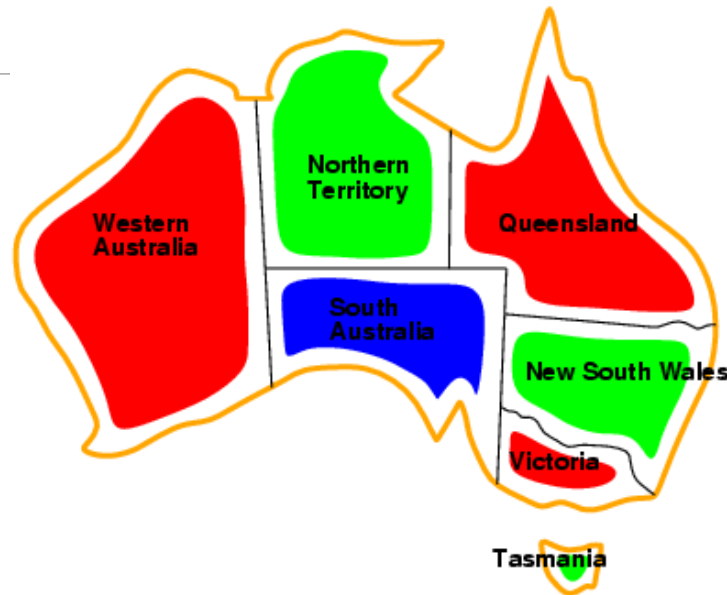
**Domains**  $D_i = \{\text{red, green, blue}\}$

**Constraints:** adjacent regions must have different colors

e.g.,  $WA \neq NT$ , or  $(WA, NT)$  in  $\{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), (\text{blue, red}), (\text{blue, green})\}$

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  $d \rightarrow O(d^n)$  complete assignments
  - e.g., Boolean CSPs, incl. 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.,  $SA \neq \text{green}$

**Binary** constraints involve pairs of variables,

- e.g.,  $SA \neq WA$

**Higher-order** constraints involve 3 or more variables,

- e.g., cryptarithmic column constraints

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

# 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
        (Xi, Xj)  $\leftarrow$  POP(queue)
        if REVISE(csp, Xi, Xj) then
            if size of Di = 0 then return false
            for each Xk in Xi.NEIGHBORS - {Xj} do
                add (Xk, Xi) to queue
    return true

function REVISE(csp, Xi, Xj) returns true iff we revise the domain of Xi
    revised  $\leftarrow$  false
    for each x in Di do
        if no value y in Dj allows (x,y) to satisfy the constraint between Xi and Xj then
            delete x from Di
        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  $k^{\text{th}}$  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 can achieve one's goals given one's beliefs (and information)
  - But what are one's goals
  - Are there 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 minimum courses does the person have to take ?
  - Q2: Can a person graduate in 2 majors studying 3 courses only?
  - ...

# Algorithms for Optimality

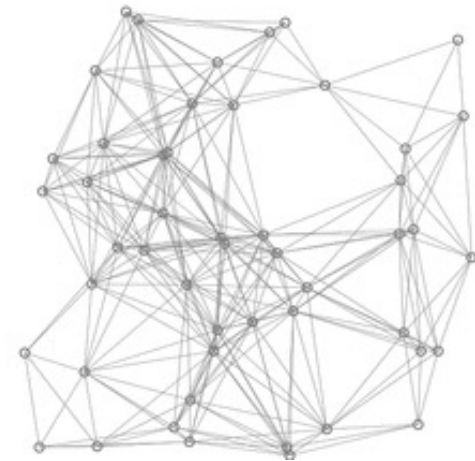
---

- 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

Source: [https://en.wikipedia.org/wiki/Dijkstra%27s\\_algorithm](https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

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



# Exercise and Code

---

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

# Course Project

---

# Project Discussion: What Problem Fascinates You ?

---

- Data
  - Water
  - Finance
  - ...
- Analytics
  - Search, Optimization, Learning, Planning, ...
- Application
  - Building chatbot
- Users
  - Diverse demographics
  - Diverse abilities
  - Multiple human languages

## Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
  - **Solving**: Choose a decision problem, identify data, work on solution methods
  - **Human interaction**: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
  - **Solving**: Evaluate your solution on problem
  - **Human interaction**: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
  - **Evaluation**: Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

# Project Discussion: Dates and Deliverables

---

## Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
  - **Solving**: Choose a decision problem, identify data, work on solution methods
  - **Human interaction**: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
  - **Solving**: Evaluate your solution on problem
  - **Human interaction**: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
  - **Evaluation**: Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

- Oct 12, 2023
  - Project checkpoint
  - In-class presentation
- Nov 30, 2023
  - Project report due
- Dec 5 / 7, 2023
  - In-class presentation

# Skeleton: A Basic Chatbot

- Run in an infinite loop until the user wants to quit
- Handle any user response
  - User can quit by typing “Quit” or “quit” or just “q”
  - User can enter any other text and the program has to handle it. The program should write back what the user entered and say – “I do not know this information”.
- Handle known user query types // Depends on your project
  - “Tell me about N-queens”, “What is N ?”
  - “Solve for N=4?”
  - “Why is this a solution? ”
- Handle chitchat // Support at least 5, extensible from a file
  - “Hi” => “Hello”
  - ...
- *Store session details in a file*

## Illustrative Project

1. **Title:** Solve and explain solving of n-queens puzzle
2. **Key idea:** Show students how a course project will look like
3. **Who will care when done:** students of the course, prospective AI students and teachers
4. **Data need:** n: the size of game; interaction
5. **Methods:** search
6. **Evaluation:** correctness of solution, quality of explanation, appropriateness of chat
7. **Users:** with and without AI background; with and without chess background
8. **Trust issue:** user may not believe in the solution, may find interaction offensive (why queens, not kings? ...)

# Project Discussion: Illustration

1. Create a private Github repository called “CSCE58x-Fall2023-<studentname>-Repo”. Share with Instructor (biplav-s) and TA (kausik-l)
2. Create Google folder called “CSCE58x-Fall2023-<studentname>-SharedInfo”. Share with Instructor ([prof.biplav@gmail.com](mailto:prof.biplav@gmail.com)) and TA ([lakkarajukausik90@gmail.com](mailto:lakkarajukausik90@gmail.com))
3. Create a Google doc in your Google repo called “Project Plan” and have the following by next class (Sep 5, 2023)

1. **Title:** Solve and explain solving of n-queens puzzle
2. **Key idea:** Show students how a course project will look like
3. **Who will care when done:** students of the course, prospective AI students and teachers
4. **Data need:** n: the size of game; interaction
5. **Methods:** search
6. **Evaluation:** correctness of solution, quality of explanation, appropriateness of chat
7. **Users:** with and without AI background; with and without chess background
8. **Trust issue:** user may not believe in the solution, may find interaction offensive (why queens, not kings? ...)

# Project Illustration: N-Queens

---

- Sprint 1: (Sep 12 – Oct 5)
  - **Solving**: Choose a decision problem, identify data, work on solution methods
    - Method 1: Random solution
    - Method 2: Search – BFS
    - Method 3: Search - ...
  - **Human interaction**: Develop a basic chatbot (no AI) as outlined
  - Deliverable
    - Code structure in Github
      - ./data
      - ./code
      - ./docs
      - ./test
    - Presentation: Make sprint presentation on Oct 12, 2023

# Reference: Project Rubric

- **Project results – 60%**
  - Working system ? – 30%
  - Evaluation with results superior to baseline? – 20%
  - Considered related work? – 10%
- **Project efforts – 40%**
  - Project report – 20%
  - Project presentation (updates, final) – 20%
- **Bonus**
  - Challenge level of problem – 10%
  - Instructor discretion – 10%
- **Penalty**
  - Lack of timeliness as per announced policy (right) - up to 30%

## Milestones and Penalties

- Oct 12, 2023
  - Project checkpoint
  - In-class presentation
  - **Penalty: presentation not ready by Oct 10, 2023 [-10%]**
- Nov 30, 2023
  - Project report due
  - **Project report not ready by date [-10%]**
- Dec 5 / 7, 2023
  - In-class presentation
  - **Project presentations not ready by Dec 4, 2023 [-10%]**



# Lecture 10: Summary

---

- We talked about
  - Constraint Satisfaction Problem
  - Optimization Problems

# Concluding Section

---

# About Next Lecture – Lecture 11

---

# Lecture 11: Machine Learning

---

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