AI2002 Artificial Intelligence

Undergraduate Course

SPRING 2022

Instructor and AI Course Coordinator

Dr. Muhammad Farrukh Shahid

Computer Science Department NUCES-FAST Karachi

Welcome to the Course

About myself

Course Description (Interesting Ingredient)

Marks Distribution

Assessment Item	Number	Weight (%)
Assignments	4	**10**
Midterm Exam	2	**15 each**
Project (Theory / Lab)	1	**10**
Final Exam	1	**50**

Teaching Material:

Textbook:

S. Russell and P. Norvig: ***Artificial Intelligence: A Modern Approach***. Pearson, 2010, *3rd Edition*

Additional Resources

AI MA

Artificial Intelligence: A Modern Approach

(Third edition) by Stuart Russell and Peter Norvig

AIMA Home
Code
Contents
Courses
Errata
Instructors

The leading textbook in Artificial Intelligence.
Used in over 1300 universities in over 110 countries.
The 22nd most cited computer science publication on Citeseer (and 4th most cited publication of this century).

What's New

- Free Online AI course, Berkeley's CS 188, offered through edX.

Comments and Discussion

- Comments from readers
- Errata list (errors in the book)
- AIMA-talk discussion list, open to all

AI Resources on the Web

- AI Resources in many categories
- AI courses that are using AIMA (1300 schools)

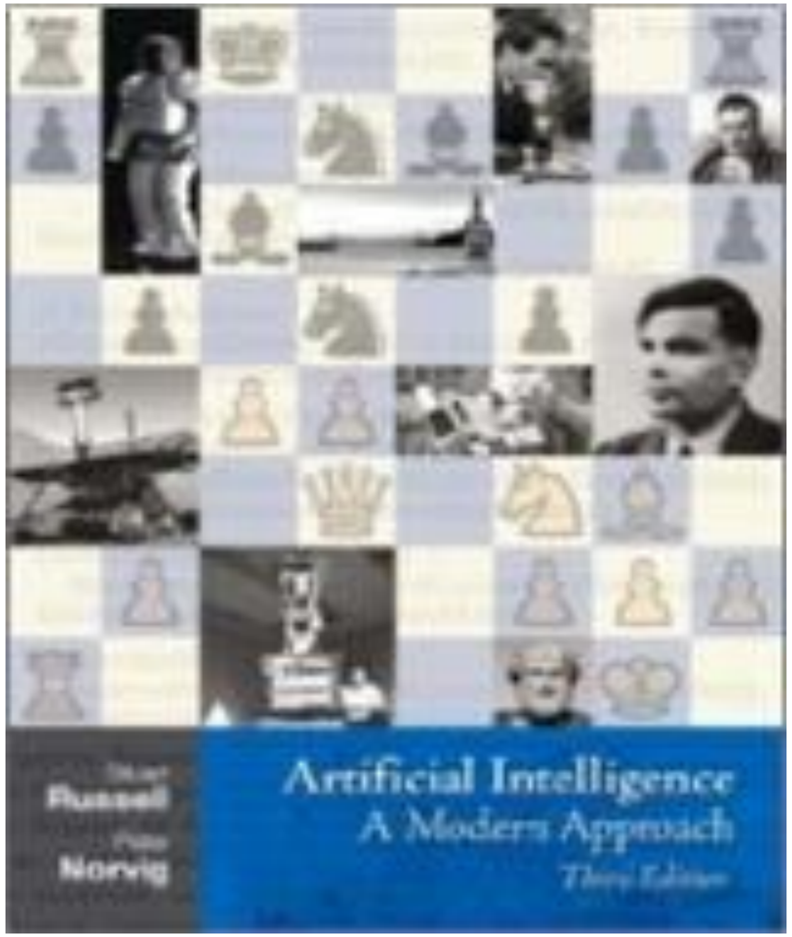
Online Code Repository

- Pseudo-code algorithms from the book in pdf.
- Online code at aimacode project on Github
- Online demos (Java applets and Javascript)
- The OpenNERO 3D multiagent simulator

For the Instructor

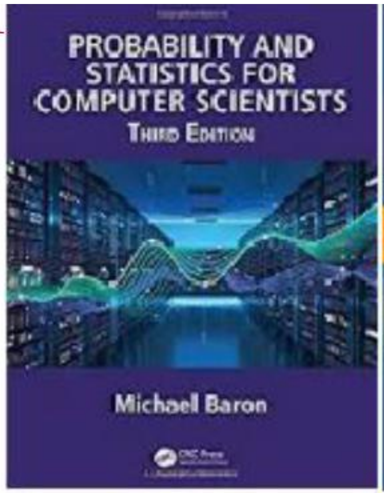
- AI Instructor's Resource Page
- Lecture slides coming soon

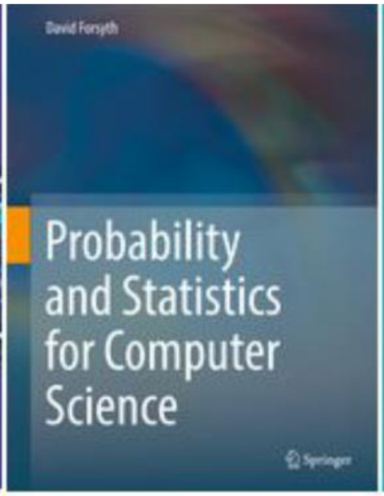
Table of Contents

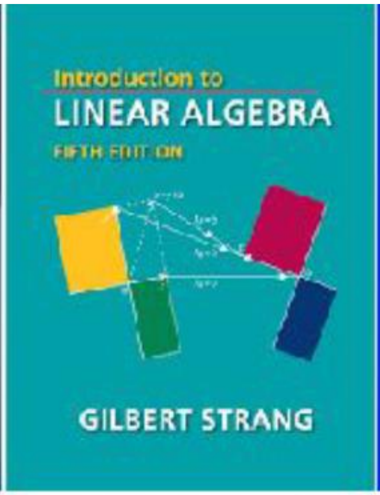


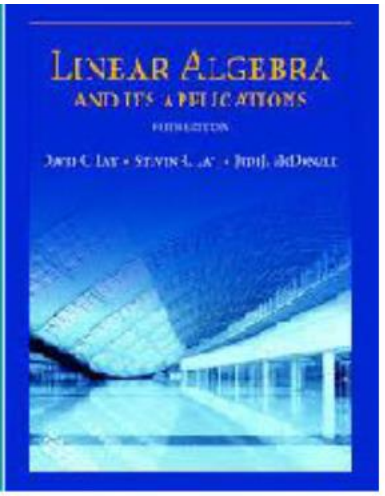
http://aima.cs.berkeley.edu/

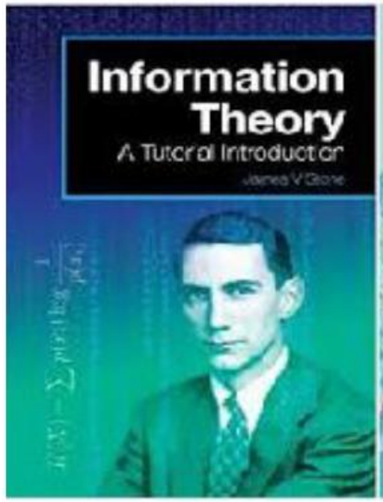
Books and Supporting Material

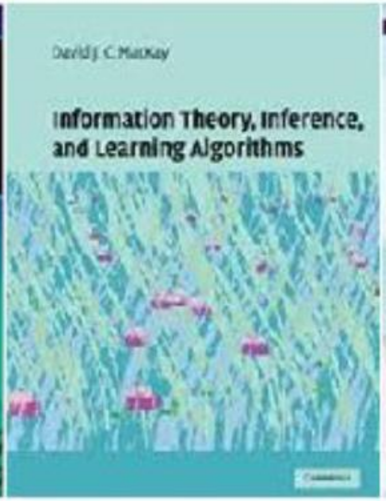







LINEAR ALGEBRA AND ITS APPLICATIONS
FIFTH EDITION
David C. Lay • Steven R. Lay • Judi J. McDonald





Reference Material / Books *(For your Future Need as well)*

1. Mathematics for Machine Learning, Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong, Cambridge University Press, 2020 ISBN-13: 978-1108455145
2. Probability and Statistics for Computer Scientists, Michael Baron, 3rd Edition (or 2nd Edition), CRC Press, 2019
3. Python for Probability, Statistics and Machine Learning, José Unpingco, Springer International, 2016
4. Probability and Statistics for Computer Science, David Forsyth, 1st Edition, Springer International, 2018
5. Basics of Linear Algebra for Machine Learning: Discover the Mathematical Language of Data in Python, Jason Brownlee, 2018
6. Introduction to Linear Algebra, Gilbert Strang, 5th Edition, Wellesley – Cambridge Press, 2016
7. Linear Algebra and its Applications, David C. Lay and Steven R. Lay, 5th Edition, Pearson Education, 2016
8. Information Theory, A tutorial Introduction, James V. Stone, 1st Edition, Sebtel Press, 2015
9. Information Theory, Inference, and Learning Algorithms, David J. C. MacKay, 4th Printing, Cambridge University Press, 2003
10. Convex Optimization, Boyd and Vandenberghe, Cambridge University Press, 2004

Tutorials, Handouts, and Scientific Research Papers

GOOGLE CLASS ROOM

BCS-6B

3piujhz

https://classroom.google.com/c/NDU4Nzk5MDU3MzU0?cjc=3piujhz

GOOGLE CLASS ROOM

BSE-6A

ec7j6ex

https://classroom.google.com/c/NDU4Nzk5MTI1MjM3?cjc=ec7j6ex

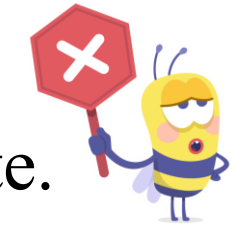
GOOGLE CLASS ROOM

BAI-4A

x3a2pvj

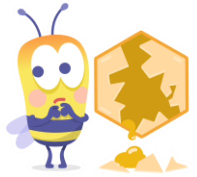
https://classroom.google.com/c/NDU4Nzk5MTI1NDMz?cjc=x3a2pvj

Class Room Policies – Pay attention



- Don’t come into the class if you are late.
- Don’t sit in my class for time passing or you have less interest.
- Attend the class with full motivation and passion.

Remember you are in the class to learn something new.



Class Room Policies

Activities such as Lecture recording and taking Photos of the board are not allowed

Course Policies – Pay attention

- Assignments must be submitted with in ***due dates***.
- Late submission will be subjected to the penalty which is as follow:

After ***2*** days of deadline ***30 %*** of marks deductions
After ***4*** days of deadline ***40 %*** of marks deductions
After ***5*** days of deadline ***100 %*** of marks deductions

- Student contact hours in my office are

Thursday 10:00 AM - 1:00 PM

Above all:

- Maintain Discipline in the class. Not even in class overall in your personality.
- May be lecture is not interesting for you. But for someone who wants to learn something so let him / her to learn.
- We have to grow as a **Nation** not individual.

Communication rules and Contact hours

- For your queries related to the course, send an email to the following email address with clearly mention your Class and Student ID in the SUBJECT of an email.

mfarrukh.shahid@nu.edu.pk

Or you can visit the office (in contacting hours) **Academic Block 3 Room no 06**

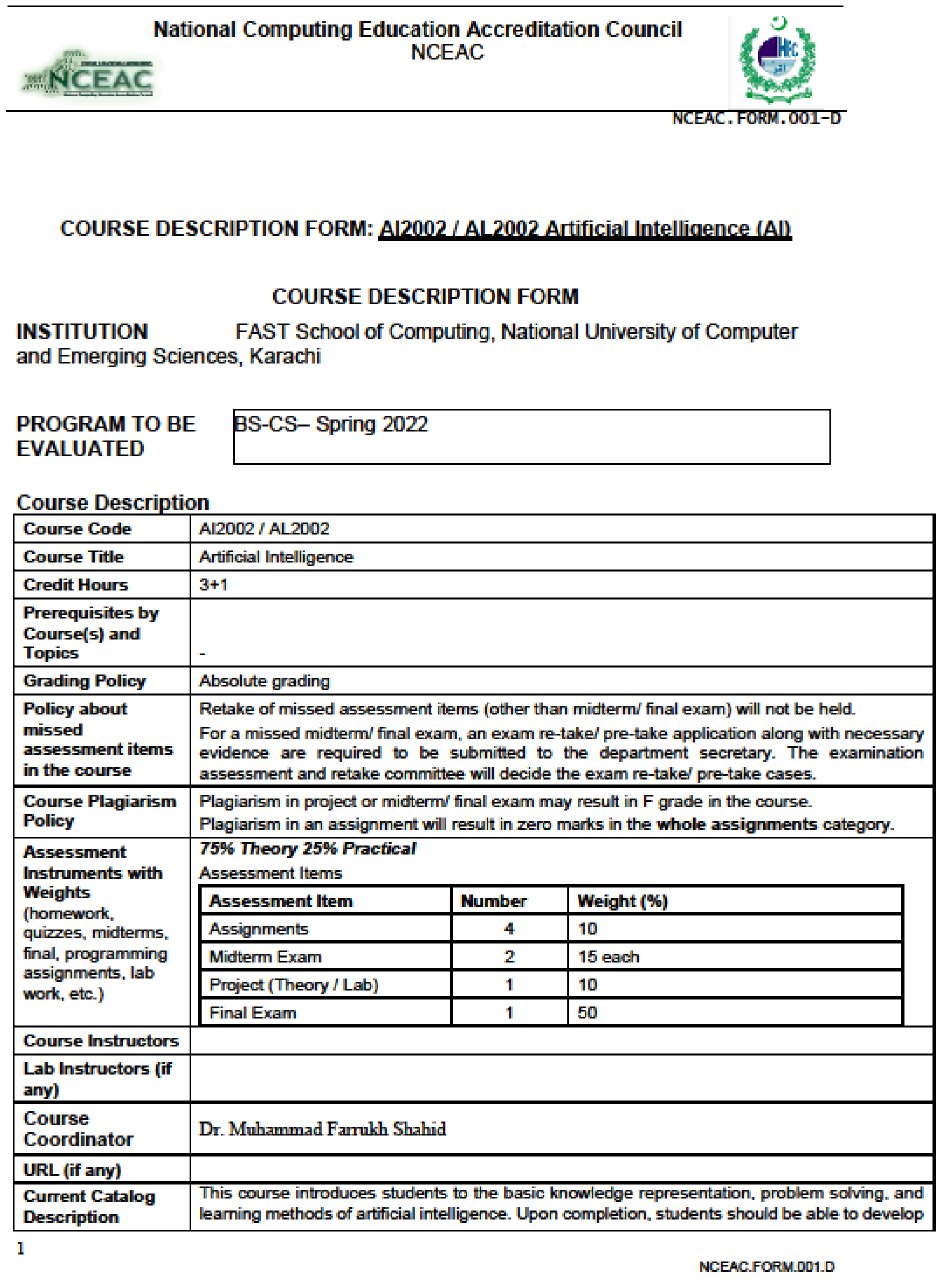
Or discuss in the class (this is highly subjected to the time availability in a class)

The ***Course*** *objectives are follows,*

Course Objectives

- How AI has been changing the world ?
- From theory to the practical – Neuron -> UAVs, Robots etc.
- Insights of Model-driven and Data-Driven approaches.
- How to introduce intelligence into the devices of IoT network, 6G mobile networks ?
- Give you full insight to chose your Final Year Project.
- Developing International Collaboration
- Roadmap for Higher Education Abroad

Course Content



National Computing Education Accreditation Council
NCEAC

NCEAC.FORM.001-D

COURSE DESCRIPTION FORM: AI2002 / AL2002 Artificial Intelligence (AI)

COURSE DESCRIPTION FORM

INSTITUTION FAST School of Computing, National University of Computer and Emerging Sciences, Karachi

PROGRAM TO BE EVALUATED BS-CS– Spring 2022

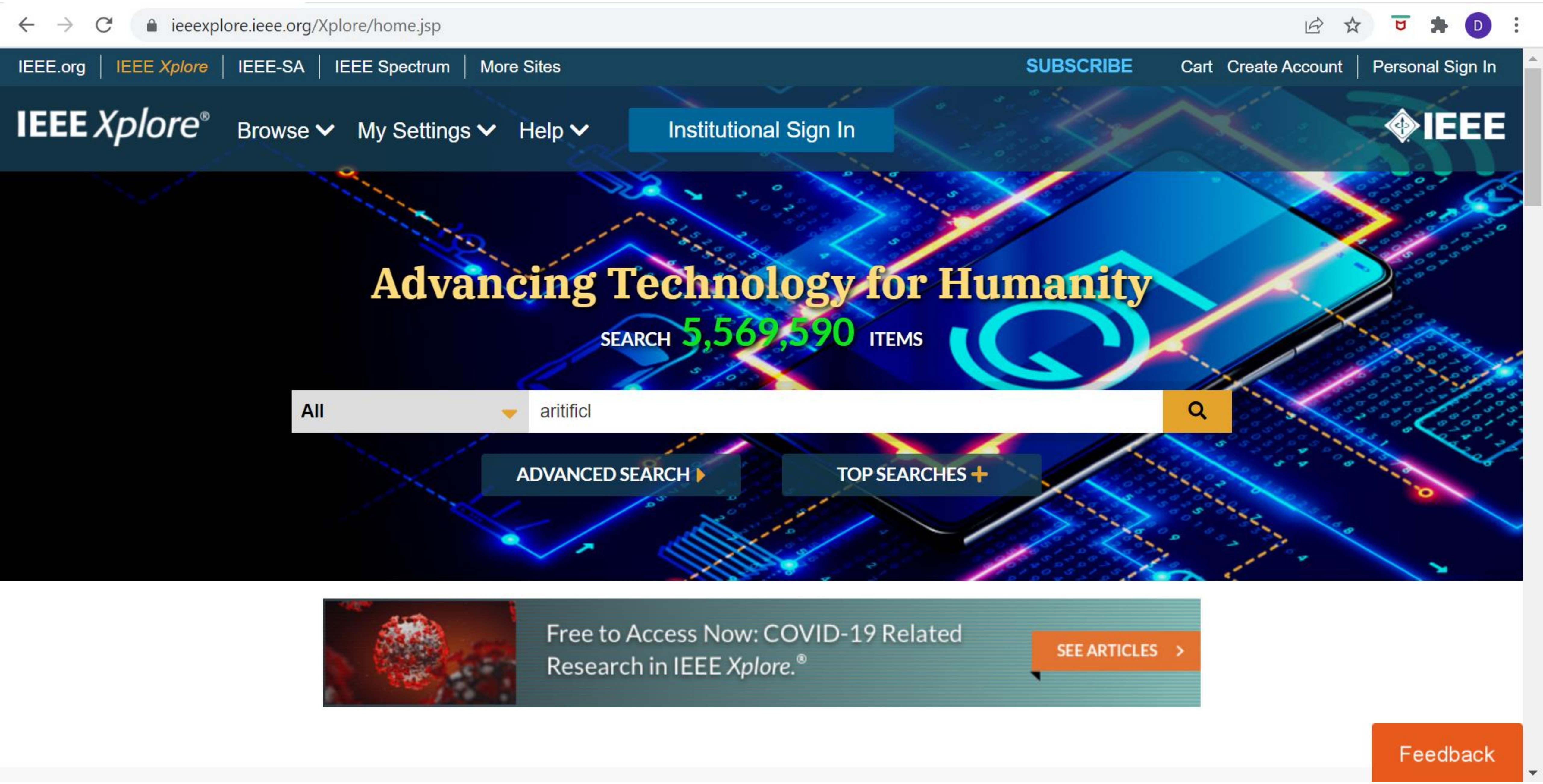
Course Description

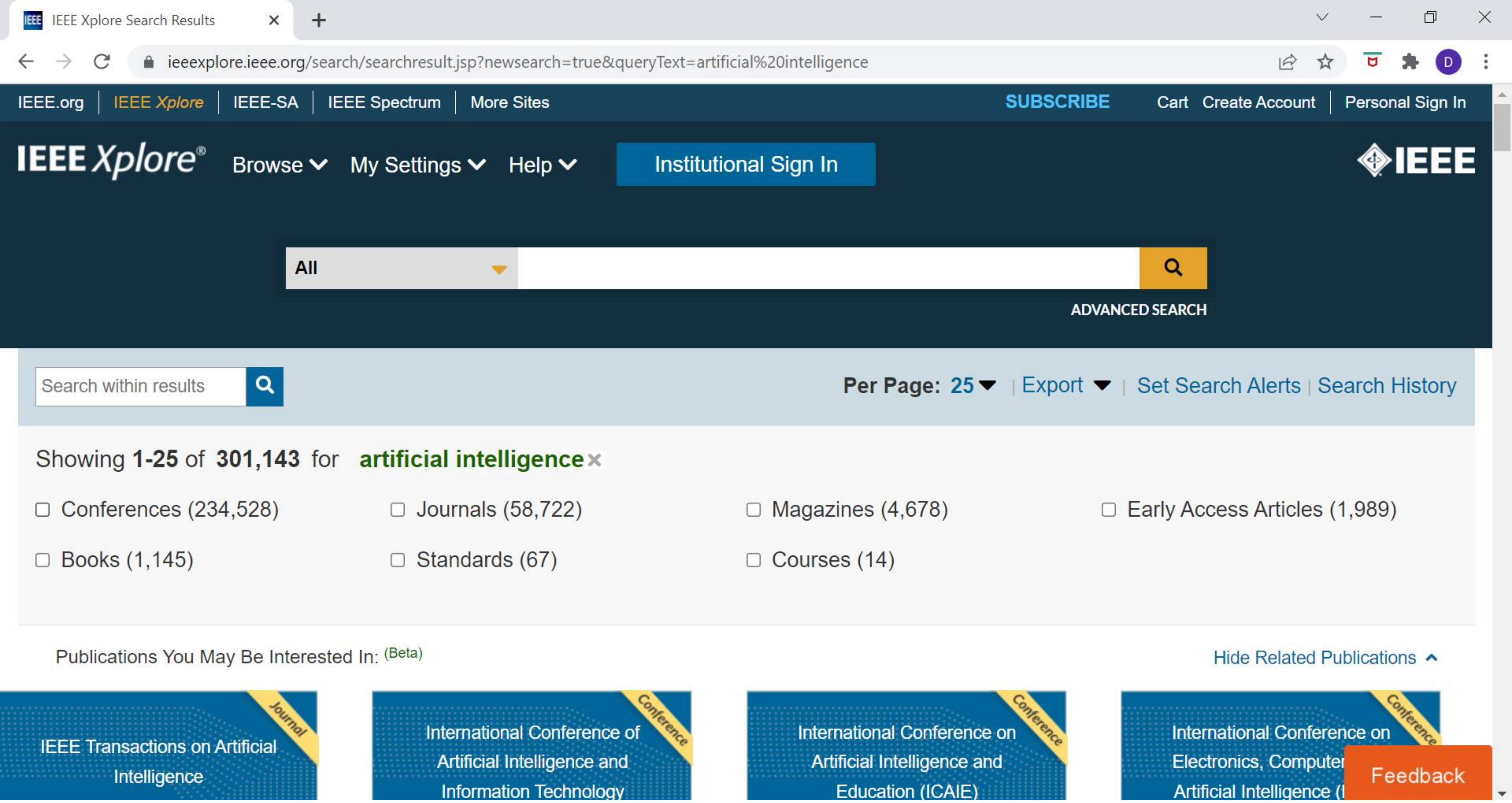
Course Code	AI2002 / AL2002
Course Title	Artificial Intelligence
Credit Hours	3+1
Prerequisites by Course(s) and Topics	-
Grading Policy	Absolute grading
Policy about missed assessment items in the course	Retake of missed assessment items (other than midterm/ final exam) will not be held. For a missed midterm/ final exam, an exam re-take/ pre-take application along with necessary evidence are required to be submitted to the department secretary. The examination assessment and retake committee will decide the exam re-take/ pre-take cases.
Course Plagiarism Policy	Plagiarism in project or midterm/ final exam may result in F grade in the course. Plagiarism in an assignment will result in zero marks in the **whole assignments** category.
Assessment Instruments with Weights (homework, quizzes, midterms, final, programming assignments, lab work, etc.)	***75% Theory 25% Practical*** Assessment Items (see table below)
Course Instructors	
Lab Instructors (if any)	
Course Coordinator	Dr. Muhammad Farrukh Shahid
URL (if any)	
Current Catalog Description	This course introduces students to the basic knowledge representation, problem solving, and learning methods of artificial intelligence. Upon completion, students should be able to develop

Assessment Item	**Number**	**Weight (%)**
Assignments	4	10
Midterm Exam	2	15 each
Project (Theory / Lab)	1	10
Final Exam	1	50

1

NCEAC.FORM.001.D


ieeexplore.ieee.org/Xplore/home.jsp
IEEE.org
IEEE Xplore
IEEE-SA
IEEE Spectrum
More Sites
SUBSCRIBE
Cart
Create Account
Personal Sign In
IEEE Xplore®
Browse
My Settings
Help
Institutional Sign In
IEEE
Advancing Technology for Humanity
SEARCH 5,569,590 ITEMS
All
aritificl
ADVANCED SEARCH
TOP SEARCHES
Free to Access Now: COVID-19 Related Research in IEEE Xplore.®
SEE ARTICLES
Feedback



Showing **1-25** of **301,143** for **artificial intelligence**

Conferences (234,528)
Journals (58,722)
Magazines (4,678)
Early Access Articles (1,989)
Books (1,145)
Standards (67)
Courses (14)

Publications You May Be Interested In: (Beta)

IEEE Transactions on Artificial Intelligence

International Conference of Artificial Intelligence and Information Technology

International Conference on Artificial Intelligence and Education (ICAIE)

International Conference on Electronics, Computer Artificial Intelligence (I

C. Tang, Z. Wang, X. Sima and L. Zhang, "Research on Artificial Intelligence Algorithm and Its Application in Games," *2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM)*, 2020, pp. 386-389, doi: 10.1109/AIAM50918.2020.00085.

Z. Li, "Analysis on the Influence of Artificial Intelligence Development on Accounting," *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2020, pp. 260-262, doi: 10.1109/ICBAIE49996.2020.00061.

Z. Yanhua, "The Application of Artificial Intelligence in Foreign Language Teaching," *2020 International Conference on Artificial Intelligence and Education (ICAIE)*, 2020, pp. 40-42, doi: 10.1109/ICAIE50891.2020.00017.

Z. Yanhua, "The Application of Artificial Intelligence in Foreign Language Teaching," *2020 International Conference on Artificial Intelligence and Education (ICAIE)*, 2020, pp. 40-42, doi: 10.1109/ICAIE50891.2020.00017.

Y. Liu and P. Tang, "The Prospect for the Application of the Surgical Navigation System Based on Artificial Intelligence and Augmented Reality," *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2018, pp. 244-246, doi: 10.1109/AIVR.2018.00056.

X. Fu, "The Application of Artificial Intelligence Technology in College Physical Education," *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2020, pp. 263-266, doi: 10.1109/ICBAIE49996.2020.00062.

F. Shi *et al*., "Review of Artificial Intelligence Techniques in Imaging Data Acquisition, Segmentation, and Diagnosis for COVID-19," in *IEEE Reviews in Biomedical Engineering*, vol. 14, pp. 4-15, 2021, doi: 10.1109/RBME.2020.2987975.

N. Zheng *et al*., "Predicting COVID-19 in China Using Hybrid AI Model," in *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 2891-2904, July 2020, doi: 10.1109/TCYB.2020.2990162.

Y. Liu and P. Tang, "The Prospect for the Application of the Surgical Navigation System Based on Artificial Intelligence and Augmented Reality," *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2018, pp. 244-246, doi: 10.1109/AIVR.2018.00056.

F. Lo, F. Su, S. Chen, J. Qiu and J. Du, "Artificial Intelligence Aided Innovation Education Based on Multiple Intelligence," *2021 IEEE International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, 2021, pp. 12-15, doi: 10.1109/ICAIRC52191.2021.9544874.

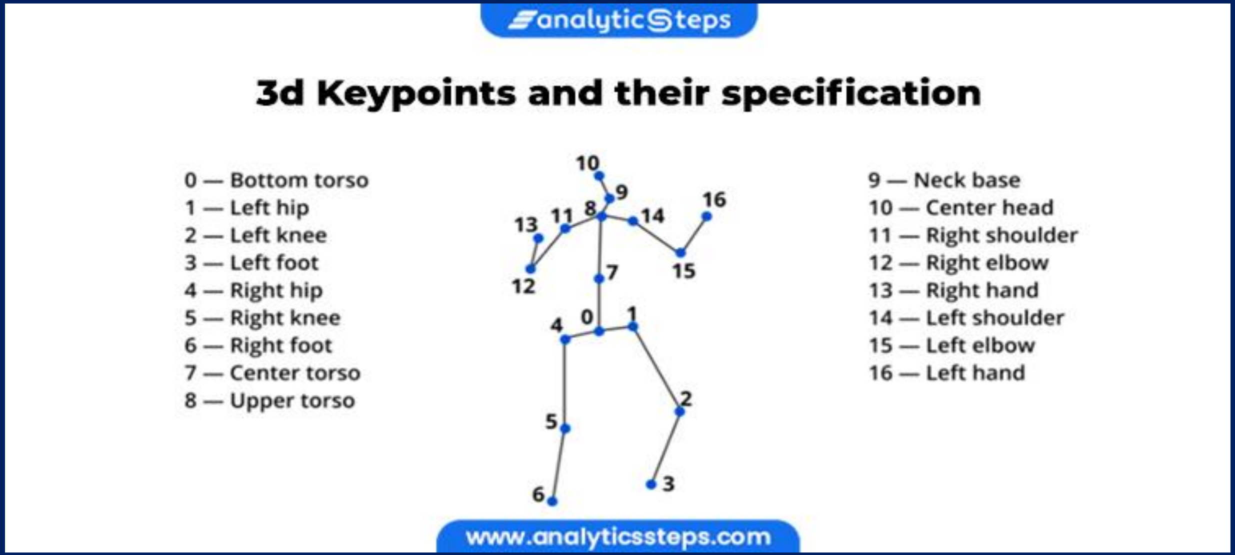
Aadhityan A, "A novel method for implementing Artificial Intelligence, Cloud and Internet of Things in Robots," *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2015, pp. 1-4, doi: 10.1109/ICIIECS.2015.7193238.

M. -F. R. Lee and T. -W. Chien, "Artificial Intelligence and Internet of Things for Robotic Disaster Response," *2020 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, 2020, pp. 1-6, doi: 10.1109/ARIS50834.2020.9205794.

Shall I start ?

https://thecowbotchannel.com/the-rise-of-artificial-intelligence-in-the-fashion-and-clothing-industry

https://www.globaltechoutlook.com/how-ai-is-powering-the-footwear-industry-today/

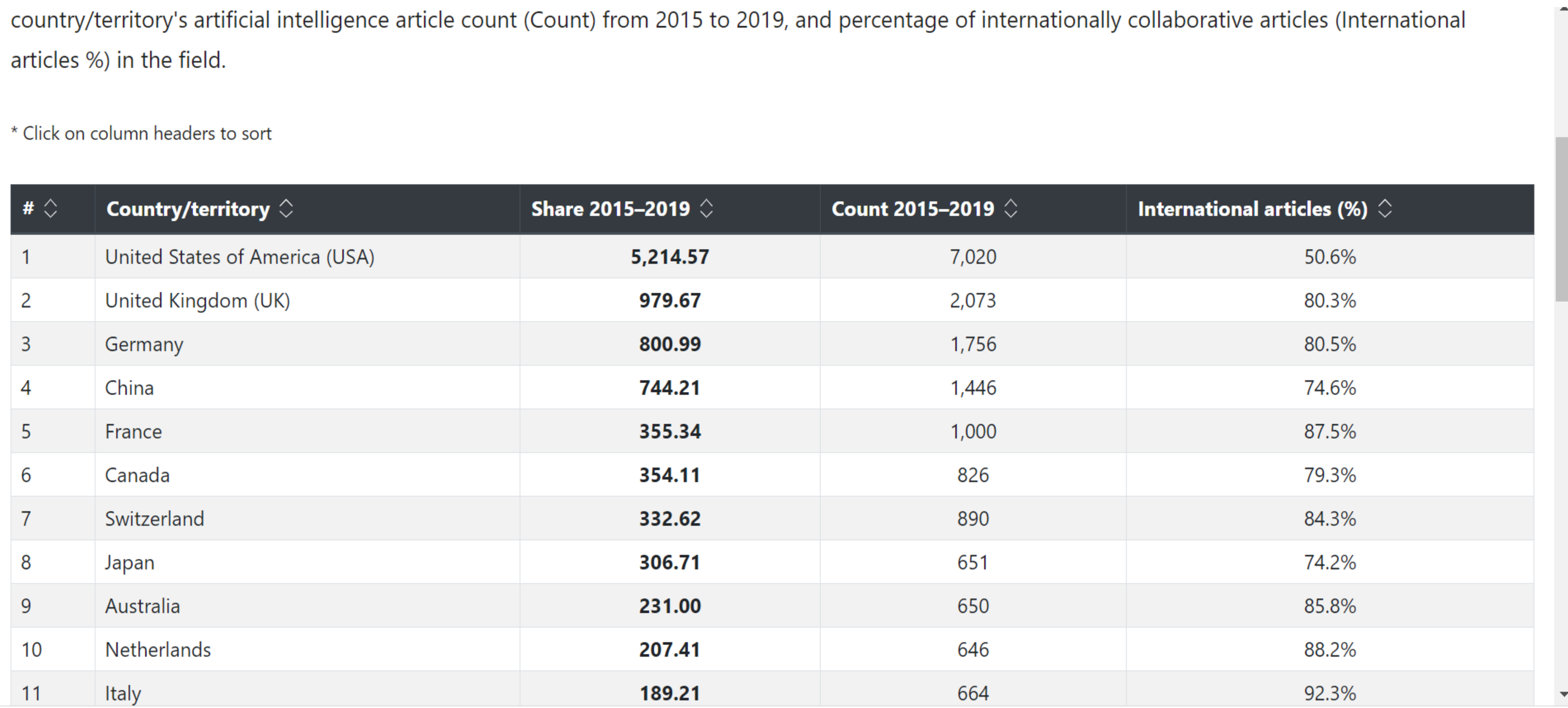


3d Keypoints and their specification

0 — Bottom torso	9 — Neck base
1 — Left hip	10 — Center head
2 — Left knee	11 — Right shoulder
3 — Left foot	12 — Right elbow
4 — Right hip	13 — Right hand
5 — Right knee	14 — Left shoulder
6 — Right foot	15 — Left elbow
7 — Center torso	16 — Left hand
8 — Upper torso	

https://www.analyticssteps.com/blogs/how-ai-revolutionizing-fitness-industry

https://thiluxan.medium.com/artificial-intelligence-and-machine-learning-in-cricket-b2a69059f08d


country/territory's artificial intelligence article count (Count) from 2015 to 2019, and percentage of internationally collaborative articles (International articles %) in the field.

* Click on column headers to sort

#	Country/territory	Share 2015–2019	Count 2015–2019	International articles (%)
1	United States of America (USA)	**5,214.57**	7,020	50.6%
2	United Kingdom (UK)	**979.67**	2,073	80.3%
3	Germany	**800.99**	1,756	80.5%
4	China	**744.21**	1,446	74.6%
5	France	**355.34**	1,000	87.5%
6	Canada	**354.11**	826	79.3%
7	Switzerland	**332.62**	890	84.3%
8	Japan	**306.71**	651	74.2%
9	Australia	**231.00**	650	85.8%
10	Netherlands	**207.41**	646	88.2%
11	Italy	**189.21**	664	92.3%

Solving Problems by Searching

Solving Problems

Problem Solving in Games- It can be done by building an AI-system to solve particular problem.

To do this, *one need to define the problem statement first* and then *generating solution* by keeping the condition in mind.

Some of the most popularly used problem solving with the help of AI are:

Cheese, Travelling Sales Man problem, N-Queen Problem,

Water Jug Problem.

Problem Searching: Searching refers to as finding information one needs.

Searching is the most commonly used technique of Problem Solving in AI.

Steps to follow for Solving Problems

1- Define the Problem

2- Analyse the Problem

3- Identification of Solution

4- Choosing the Solution

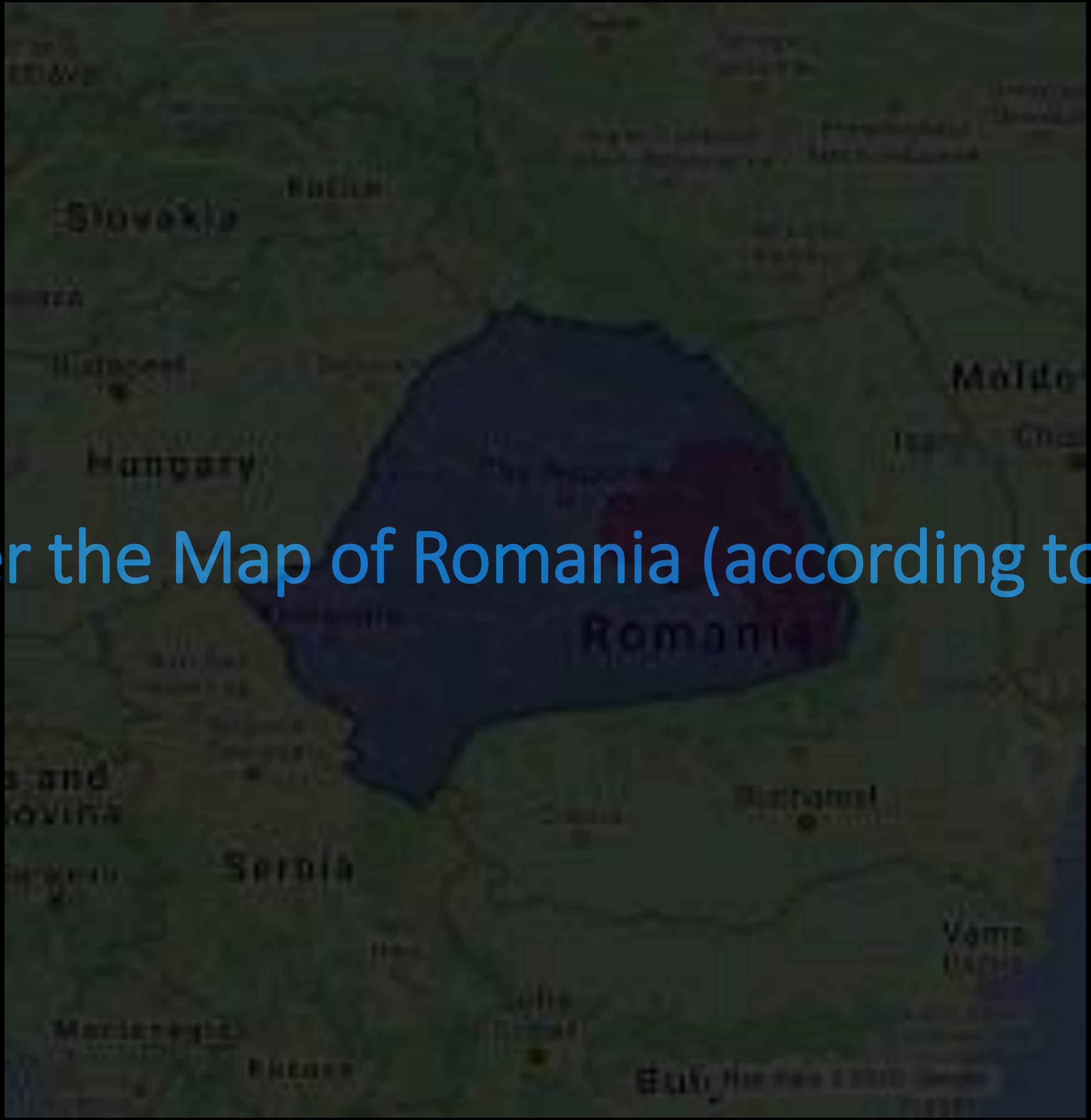
5- Implementation

Problem Solving Agent

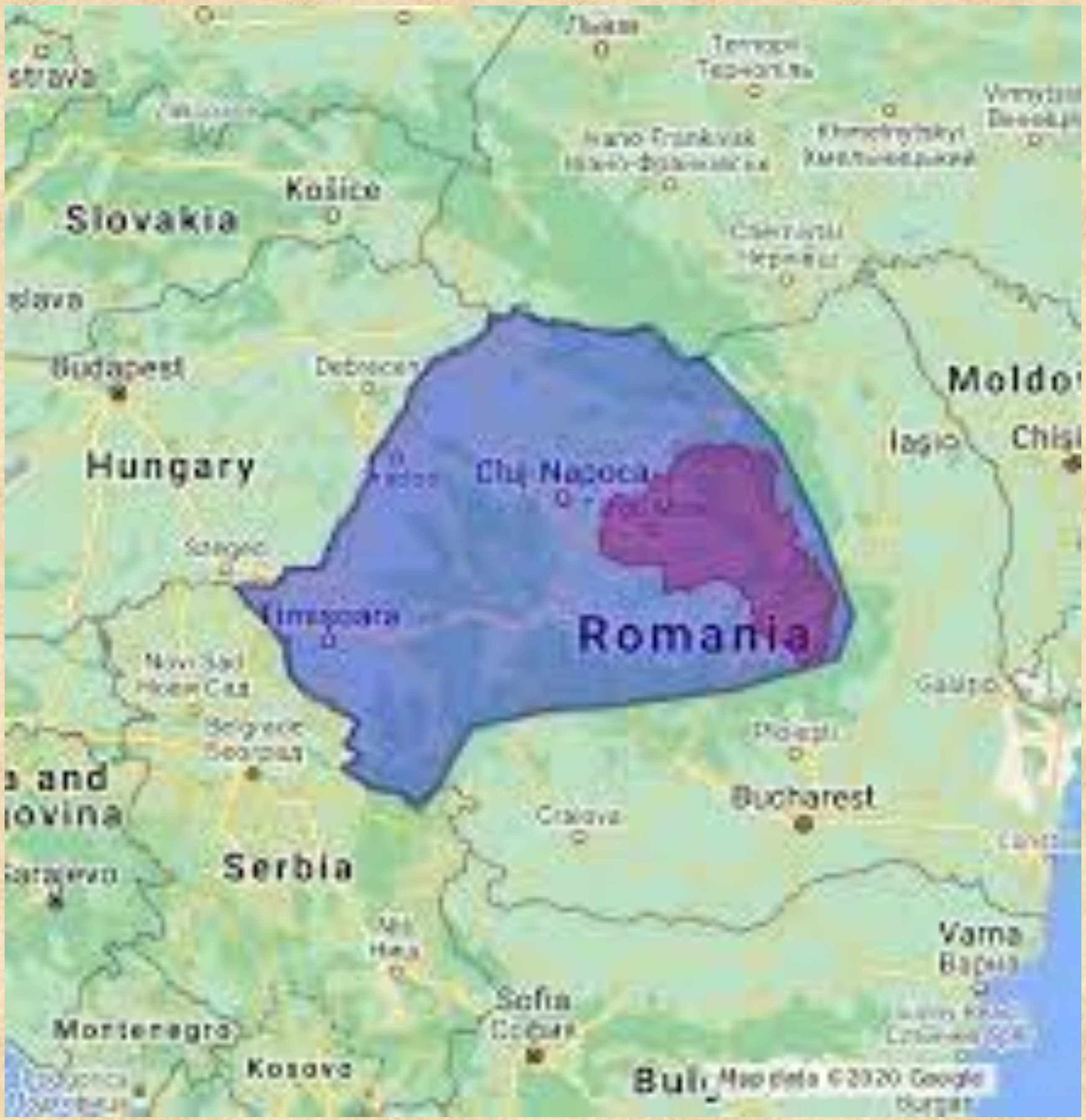
When the correct action to take is not immediately obvious, an agent may need to to plan ahead:

Consider a sequence of actions that form a path to a goal state.

Such an agent is called a **Problem-Solving Agent**, and the computational process it undertakes is called **Search**.

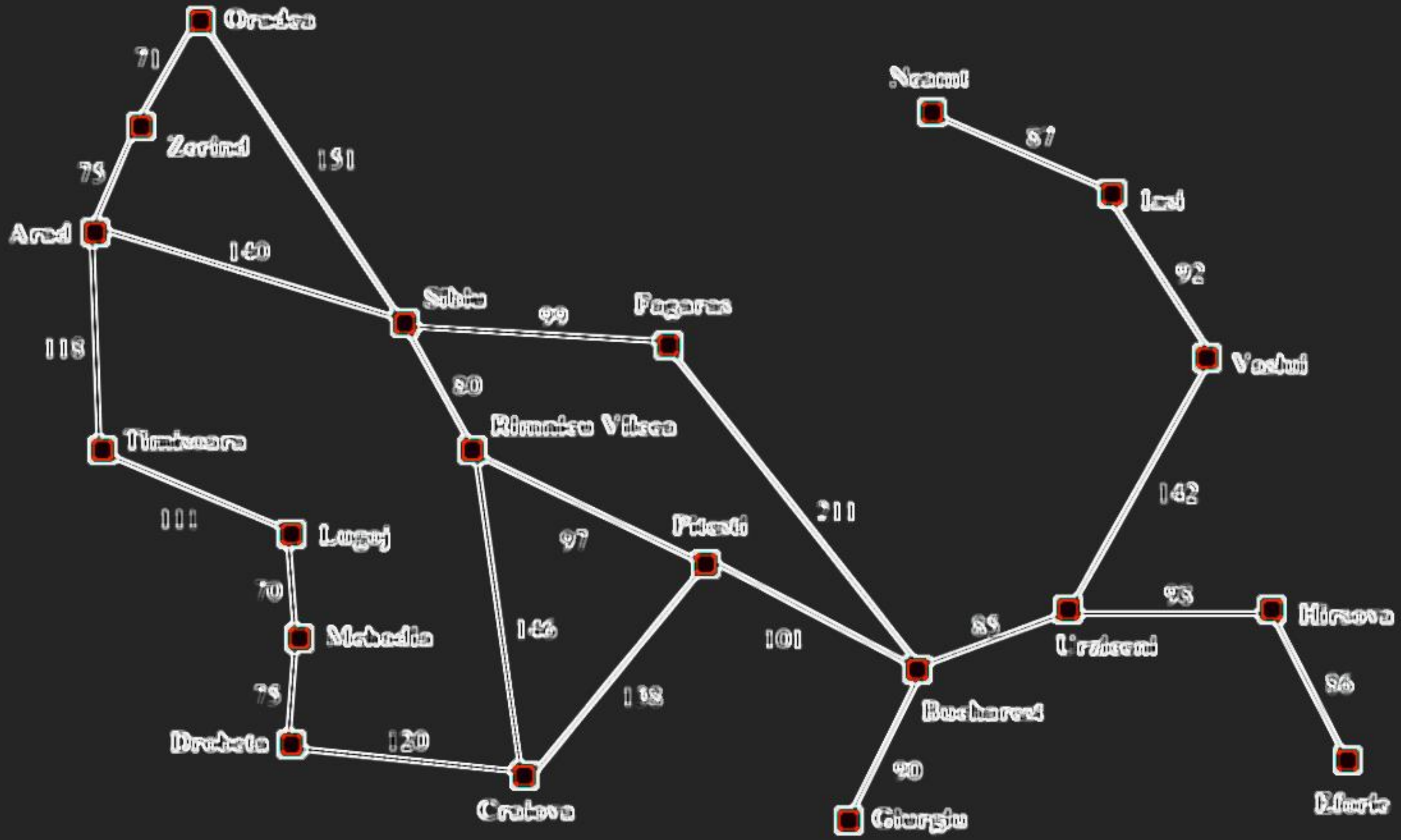


We will refer the Map of Romania (according to the Text Book)

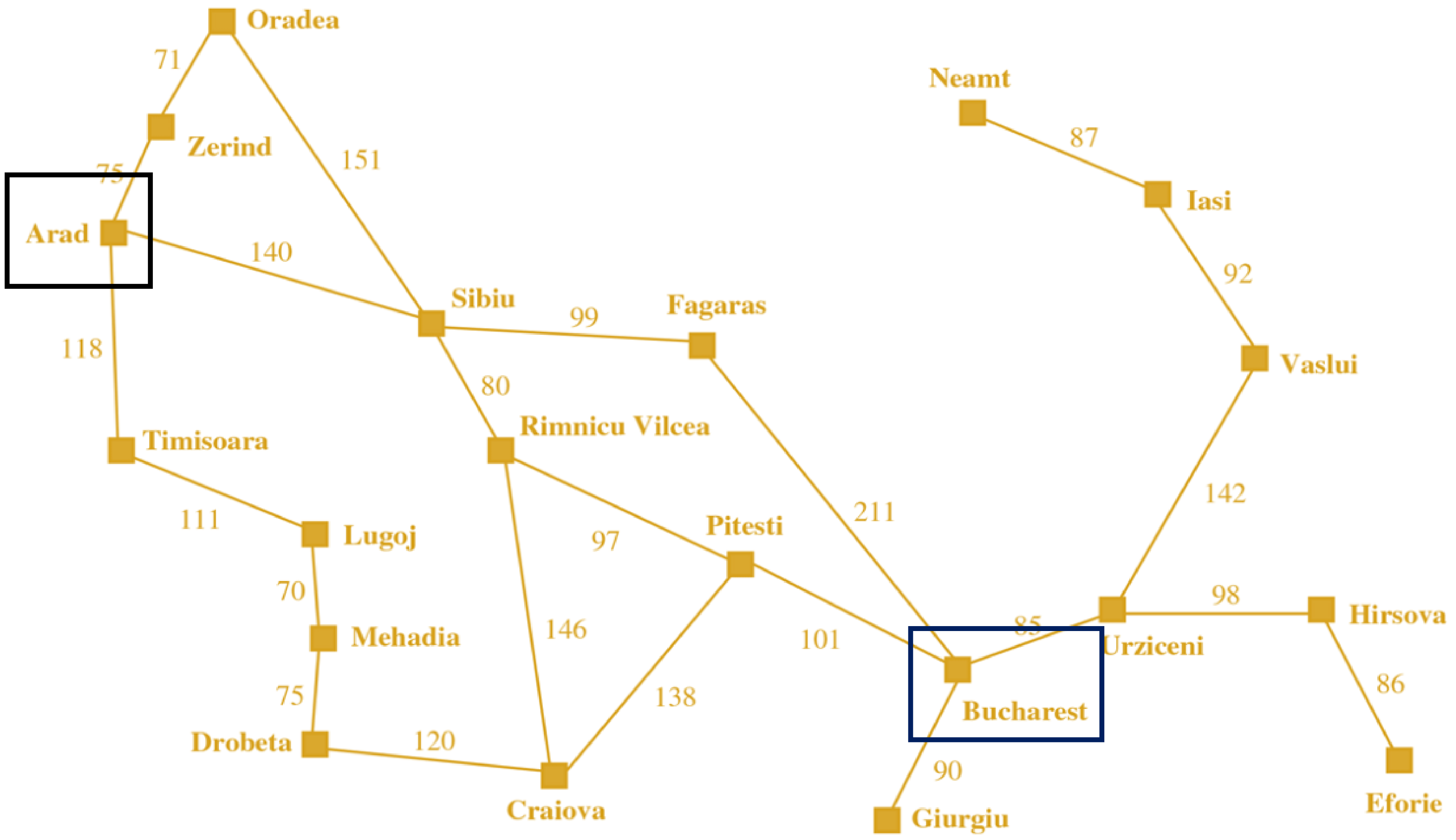

Slovakia
Košice
Budapest
Debrecen
Hungary
Cluj-Napoca
Szeged
Timișoara
Romania
Iași
Moldova
Chișinău
Novi Sad
Belgrade
Serbia
Craiova
Bucharest
Ploiești
Galați
Varna
Sofia
Montenegro
Kosovo
Sarajevo
Ivano-Frankivsk
Khmelnytskyi
Ternopil
Chernivtsi
Map data ©2020 Google

Palace of Culture, Iasi, Romania

The Red Ravine, Sebes, Romania

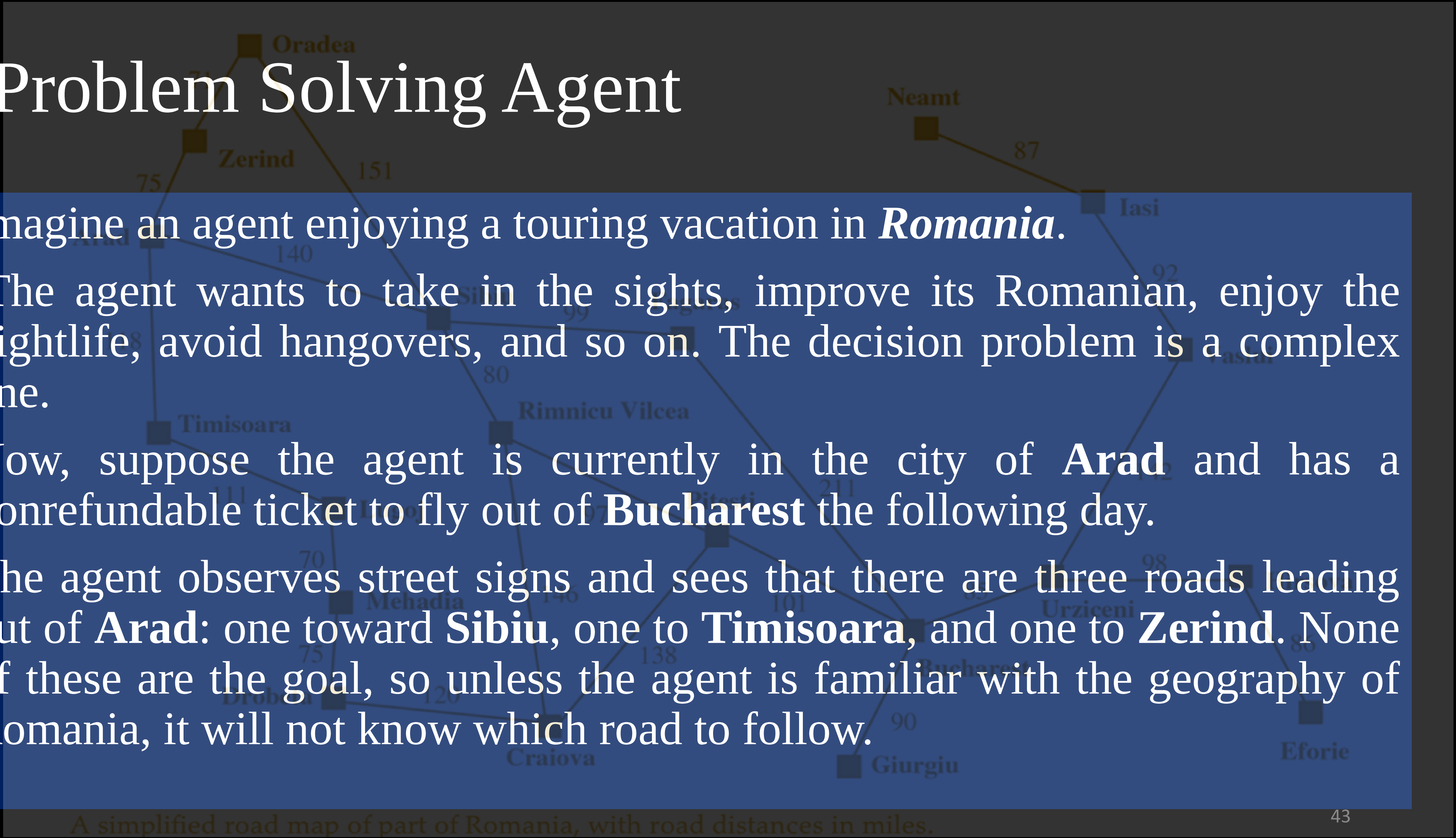


A simplified road map of part of Romania, with road distances in miles.



A simplified road map of part of Romania, with road distances in miles.

Problem Solving Agent



- Imagine an agent enjoying a touring vacation in ***Romania***.
- The agent wants to take in the sights, improve its Romanian, enjoy the nightlife, avoid hangovers, and so on. The decision problem is a complex one.
- Now, suppose the agent is currently in the city of **Arad** and has a nonrefundable ticket to fly out of **Bucharest** the following day.
- The agent observes street signs and sees that there are three roads leading out of **Arad**: one toward **Sibiu**, one to **Timisoara**, and one to **Zerind**. None of these are the goal, so unless the agent is familiar with the geography of Romania, it will not know which road to follow.

A simplified road map of part of Romania, with road distances in miles.

Problem Solving Agent

- Imagine an agent enjoying a touring vacation in ***Romania***.
- The agent wants to take in the sights, improve its Romanian, enjoy the nightlife, avoid hangovers, and so on. The decision problem is a complex one.
- Now, suppose the agent is currently in the city of **Arad** and has a nonrefundable ticket to fly out of **Bucharest** the following day.
- The agent observes street signs and sees that there are three roads leading out of **Arad**: one toward **Sibiu**, one to **Timisoara**, and one to **Zerind**. None of these are the goal, so unless the agent is familiar with the geography of Romania, it will not know which road to follow.

Problem Solving Agent

- If the agent has no additional information—that is, if the environment is unknown—then the agent can do no better than to execute one of the actions at random.
- With that information, the agent can follow this **four-phase** problem-solving process:

1 GOAL FORMULATION:

The agent adopts the goal of reaching **Bucharest**. Goals organize behavior by limiting the objectives and hence the actions to be considered.

2 PROBLEM FORMULATION:

The agent devises a description of the states and actions necessary to reach the goal- an abstract model of the relevant part of the world. For our agent, one good model is to consider the actions of traveling from one city to an adjacent city, and therefore the only fact about the state of the world that will change due to an action is the current city.

3 SEARCH:

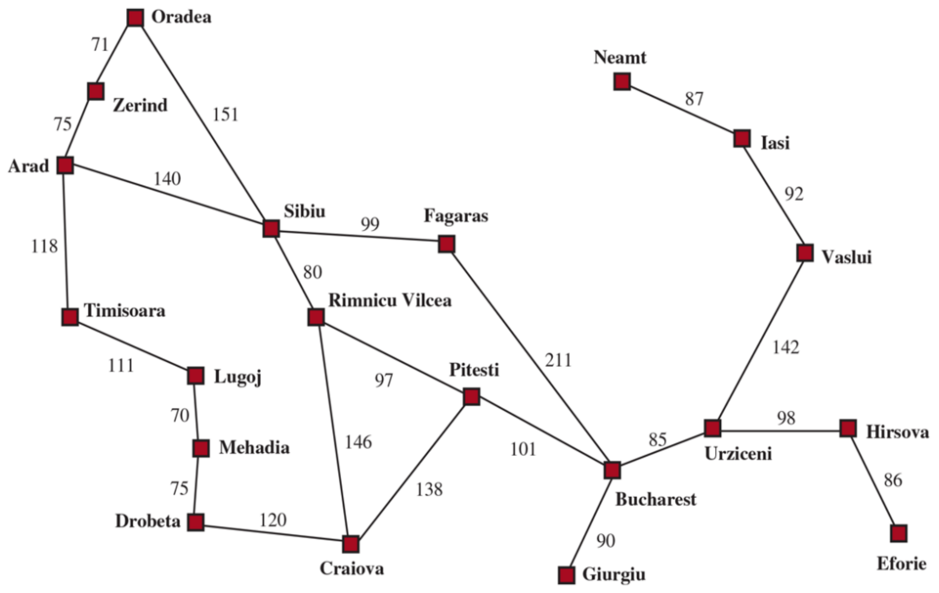
Before taking any action in the real world, the agent simulates sequences of actions in its model, searching until it finds a sequence of actions that reaches the goal. Such a sequence is called a solution. The agent might have to simulate multiple sequences that do not reach the goal, but eventually it will find a solution (such as going from **Arad to Sibiu to Fagaras to Bucharest**), or it will find that no solution is possible.

4 EXECUTION:

The agent can now execute the actions in the solution, one at a time.

It is an important property that in a fully observable, deterministic, known environment, the solution to any problem is a fixed sequence of actions: **drive to Sibiu, then Fagaras, then Bucharest**.

If the model is correct, then once the agent has found a solution, it can ignore its percepts while it is executing the actions—closing its eyes, so to speak—because the solution is guaranteed to lead to the goal.



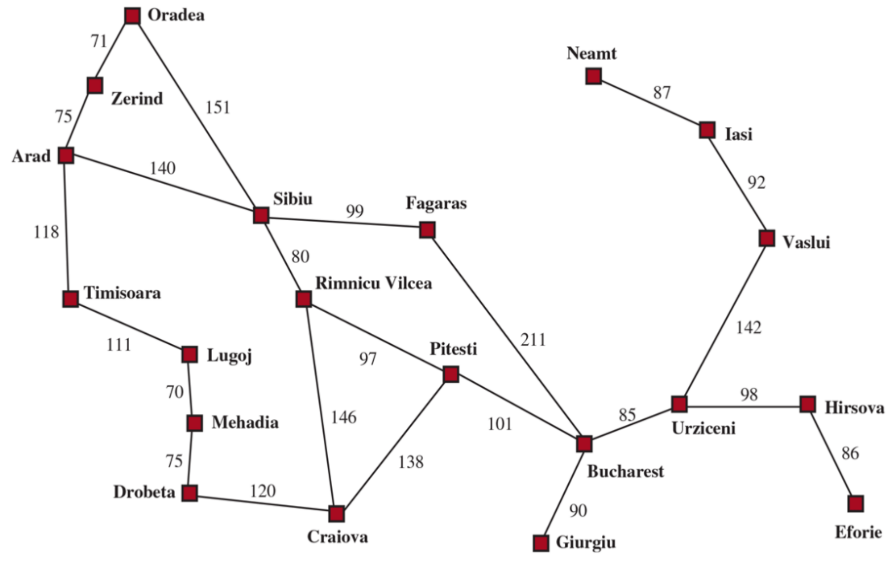
A simplified road map of part of Romania, with road distances in miles.

***Open-loop system**: ignoring the percepts breaks the loop between agent and environment*

*If there is a chance that the model is incorrect, or the environment is nondeterministic, then the agent would be safer using a **closed-loop** approach that monitors the percepts.*

In partially observable or nondeterministic environments, a solution would be a branching strategy that recommends different future actions depending on what percepts arrive.

For example, the agent might plan to drive from **Arad to Sibiu** but might need a contingency plan in case it arrives in **Zerind** by accident or finds a sign saying **“Drum Închis” (Road Closed)**



A simplified road map of part of Romania, with road distances in miles.

A search problem can be defined formally as follows:

A set of possible **states** that the environment can be in. We call this the **state space**.

The **initial state** that the agent starts in. For example: Arad

A set of one or more **goal states**. Sometimes there is **one goal state** (e.g., Bucharest), sometimes there is a small set of **alternative goal states**, and sometimes the goal is defined by a property that applies to many states (potentially an infinite number)

The **actions** available to the agent

ACTIONS(Arad) = {ToSibiu, ToTimisoara, ToZerind}.

A **transition model**, which describes what each action does. RESULT returns the state that results from doing action in state. For example

RESULT(Arad, ToZerind) = Zerind.

An **action cost function**, denoted by ACTION-COST***(s,a,s')*** when we are programing or ***c (s,a,s'),*** when we are doing math, that gives the numeric cost of applying action a in state s to reach state s'.

A problem-solving agent should use a cost function that reflects its own performance measure; for example, for route-finding agents, the cost of an action might be the length in miles, or it might be the time it takes to complete the action.

Path and a Solution

A sequence of actions forms a **path**, and a **solution** is a path from the **initial state** to a **goal state**.

We assume that action costs are additive; that is, the total cost of a path is the sum of the individual action costs. An optimal solution has the lowest path cost among all solutions.

Example Problems

A **standardized problem** is intended to illustrate or exercise various problem-solving methods. It can be given a concise, exact description and hence is suitable as a benchmark for **researchers** to compare the performance of algorithms.

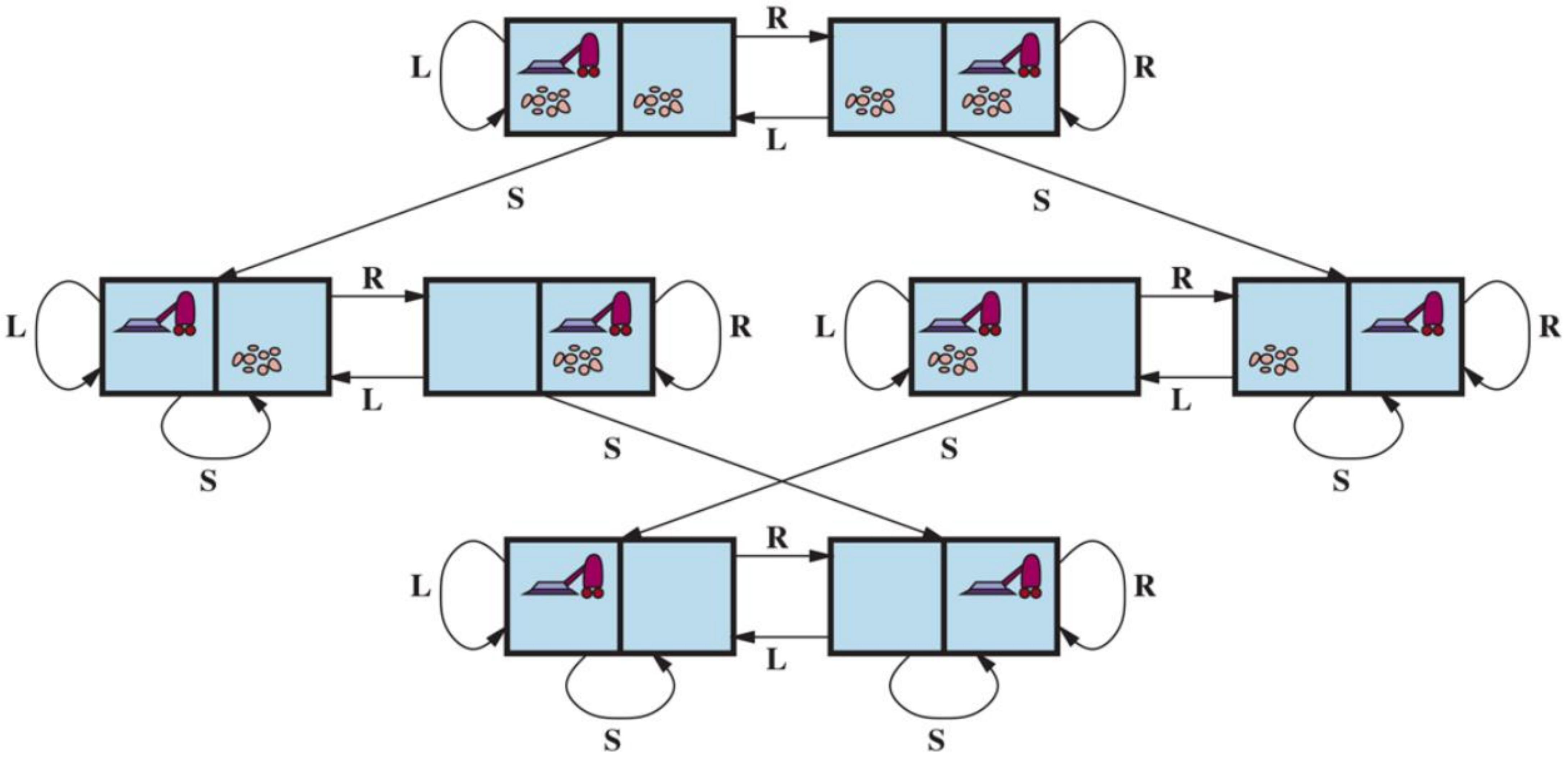
A **real-world problem**, such as robot navigation, is one whose solutions people use, and whose formulation is idiosyncratic, not standardized, because, for example, each robot has different sensors that produce different data.

Grid World (**Standardized Problem)**

A **grid world** problem is a two-dimensional rectangular array of square cells in which agents can move from cell to cell. Typically, the agent can move to any obstacle-free adjacent cell— horizontally or vertically and in some problems diagonally.

Cells can contain objects, which the agent can pick up, push, or otherwise act upon; a wall or other impassible obstacle in a cell prevents an agent from moving into that cell.

The **vacuum world** can be formulated as a grid world problem as follows:



The state-space graph for the two-cell vacuum world. There are 8 states and three actions for each state: L = $Left$, R = $Right$, S = $Suck$.

STATES: A state of the world says which objects are in which cells. For the vacuum world, the objects are the agent and any dirt. In the simple two-cell version, the agent can be in either of the two cells, and each call can either contain dirt or not, so there are states 8 states (see Figure on previous slide).

INITIAL STATE: Any state can be designated as the initial state.

ACTIONS: In the two-cell world we defined three actions: Suck, move Left, and move Right. In a two-dimensional multi-cell world, we need more movement actions. We could add Upward and Downward, giving us four absolute movement actions, or we could switch to **egocentric actions**, defined relative to the viewpoint of the agent—for example, Forward, Backward, TurnRight, and TurnLeft.

TRANSITION MODEL: Suck removes any dirt from the agent's cell; Forward moves the agent ahead one cell in the direction it is facing, unless it hits a wall, in which case the action has no effect. Backward moves the agent in the opposite direction, while Turn Right and Turn Left change the direction it is facing by 90^0

GOAL STATES: The states in which every cell is clean.

ACTION COST: Each action costs 1.

Have a look on Sokoban Puzzle Grid world problem

Real World Problem

Much more challenging.

Diverse in nature.

Continuing evolving.

Consider Airline Travel problems

STATES: Each state obviously includes a location (e.g., an airport) and the current time. Furthermore, because the cost of an action (a flight segment) may depend on previous segments, their fare bases, and their status as domestic or international, the state must record extra information about these "historical" aspects

INITIAL STATE: The user's home airport.

ACTIONS: Take any flight from the current location, in any seat class, leaving after the current time, leaving enough time for within-airport transfer if needed.

TRANSITION MODEL: The state resulting from taking a flight will have the flight's destination as the new location and the flight's arrival time as the new time.

GOAL STATE: A destination city. Sometimes the goal can be more complex, such as "arrive at the destination on a nonstop flight."

ACTION COST: A combination of monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane, frequent-flyer reward points, and so on.

Develop Model for Robot Navigation

Develop Model for UAV operating in Agriculture field for crop monitoring

Search Algorithms

A **search algorithm** takes a search problem as input and returns a solution, or an indication of failure.

We consider algorithms that superimpose a search tree over the **state-space graph**, forming various paths from the **initial state**, trying to find a path that reaches a **goal state.**

Each **node** in the search tree corresponds to a state in the state space and the edges in the search tree corresponds to actions.

The **root** of the tree corresponds to the **initial state** of the problem.

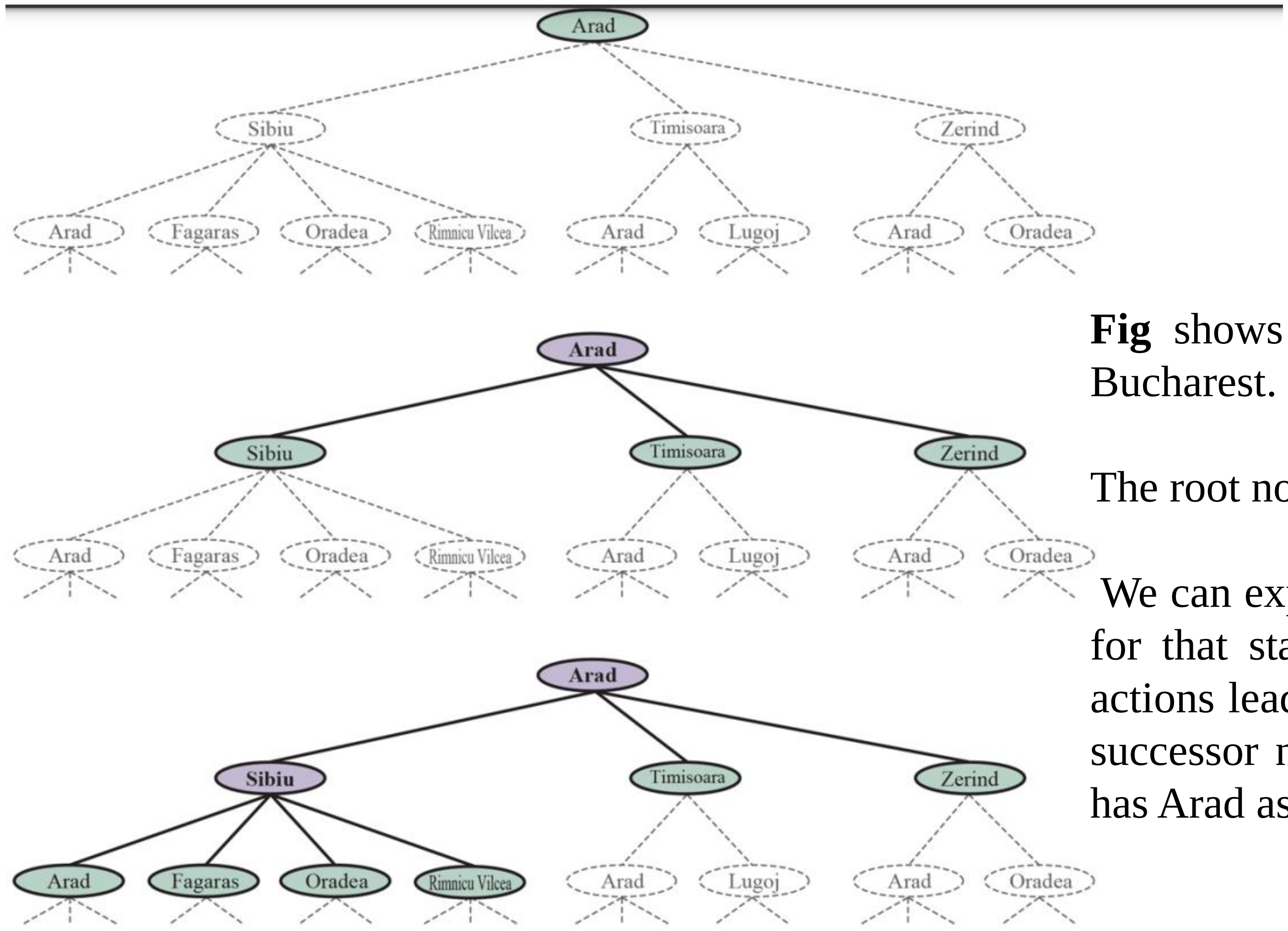
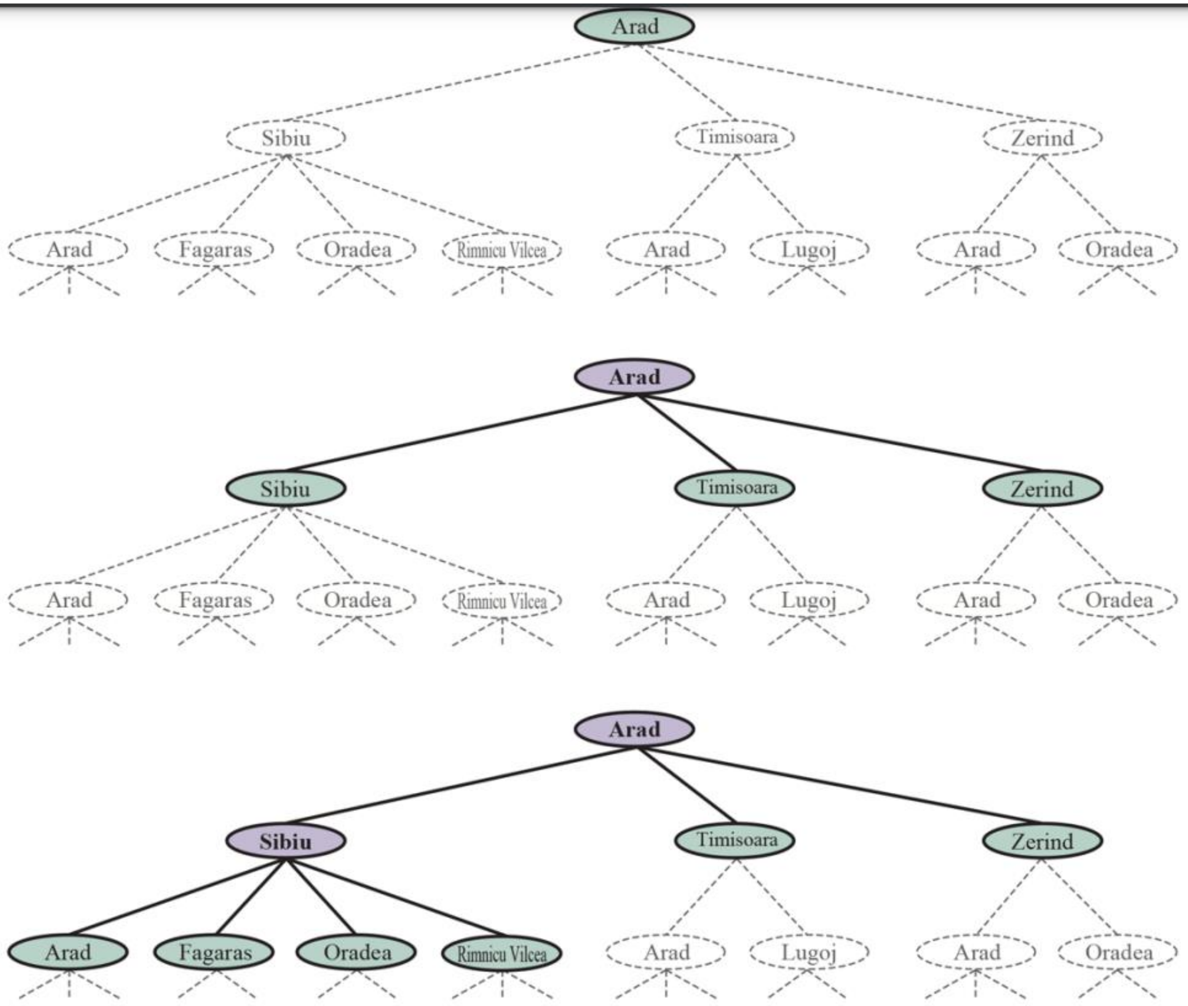


Fig shows the first few steps in finding a path from Arad to Bucharest.

The root node of the search tree is at the **initial state**, **Arad**.

We can expand the node, by considering the available **ACTIONS** for that state, using the **RESULT function** to see where those actions lead to, and generating a new node (called a child node or successor node) for each of the resulting states. Each child node has Arad as its parent node.



Now we must choose which of these three child nodes to consider next.
This is the essence of **search**—following up one option now and putting the others aside for later.
Suppose we choose to expand **Sibiu first**. Figure (bottom) shows the result: a set of **6** unexpanded nodes (outlined in bold).

We call this the **frontier** of the search tree.

We say that any state that has had a node generated for it has been reached (whether that node has been expanded).

What is the difference between Graph-Search and Tree-Like Search ?

Measuring problem-solving performance

COMPLETENESS: Is the algorithm guaranteed to find a solution when there is one, and to correctly report failure when there is not?

COST OPTIMALITY Does it find a solution with the lowest path cost of all solutions?

TIME COMPLEXITY : How long does it take to find a solution? This can be measured in seconds, or more abstractly by the number of states and actions considered.

SPACE COMPLEXITY: How much memory is needed to perform the search?

Time and space complexity are considered with respect to some measure of the problem difficulty.

In theoretical computer science, the typical measure is the size of the state-space graph $|\boldsymbol{V} + \boldsymbol{E}|$, where V is the number of vertices (state nodes) of the graph and E is the number of edges (distinct state/action pairs).

This is appropriate when the graph is an explicit data structure, such as the map of Romania. But in many AI problems, the graph is represented only implicitly by the initial state, actions, and transition model.

For an implicit state space, complexity can be measured in terms of ***d***, the depth or number of actions in an optimal solution; the maximum number of actions in any path; and the ***branching factor*** or number of successors of a node that need to be considered.

Types of Searching

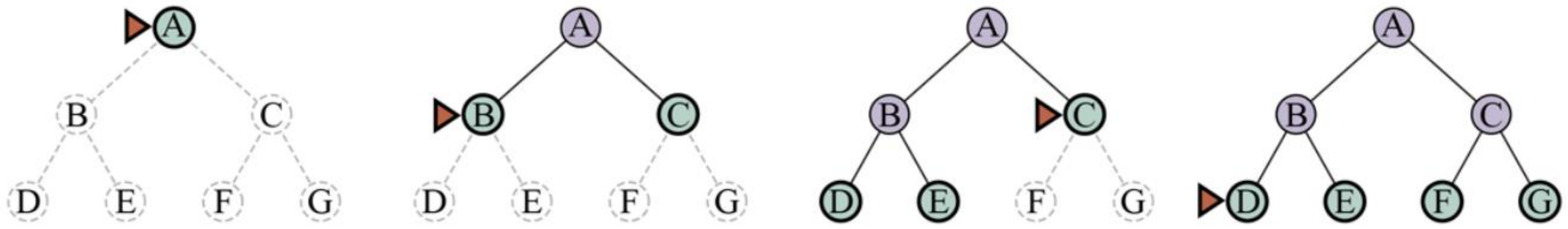
Uninformed Search

Informed Search

Uninformed Search	Informed Search
Breadth First Search	Best First Search
Uninformed Cost Search	A* Search
Depth First Search	AO* Search
Depth Limited Search	Problem Reduction
Iterative Deeping depth search	Hill climbing
Bi-directional Search	

Breadth-First Search

- Most common search algorithm.
- Search breadth-wise in a tree.
- BFS starts searching from the root node of the tree and expand all the successor node at the current level before moving to node of next level.
- BFS is an example of General-graph search algorithm.
- BFS is implemented using *FIFO* queue data structure.



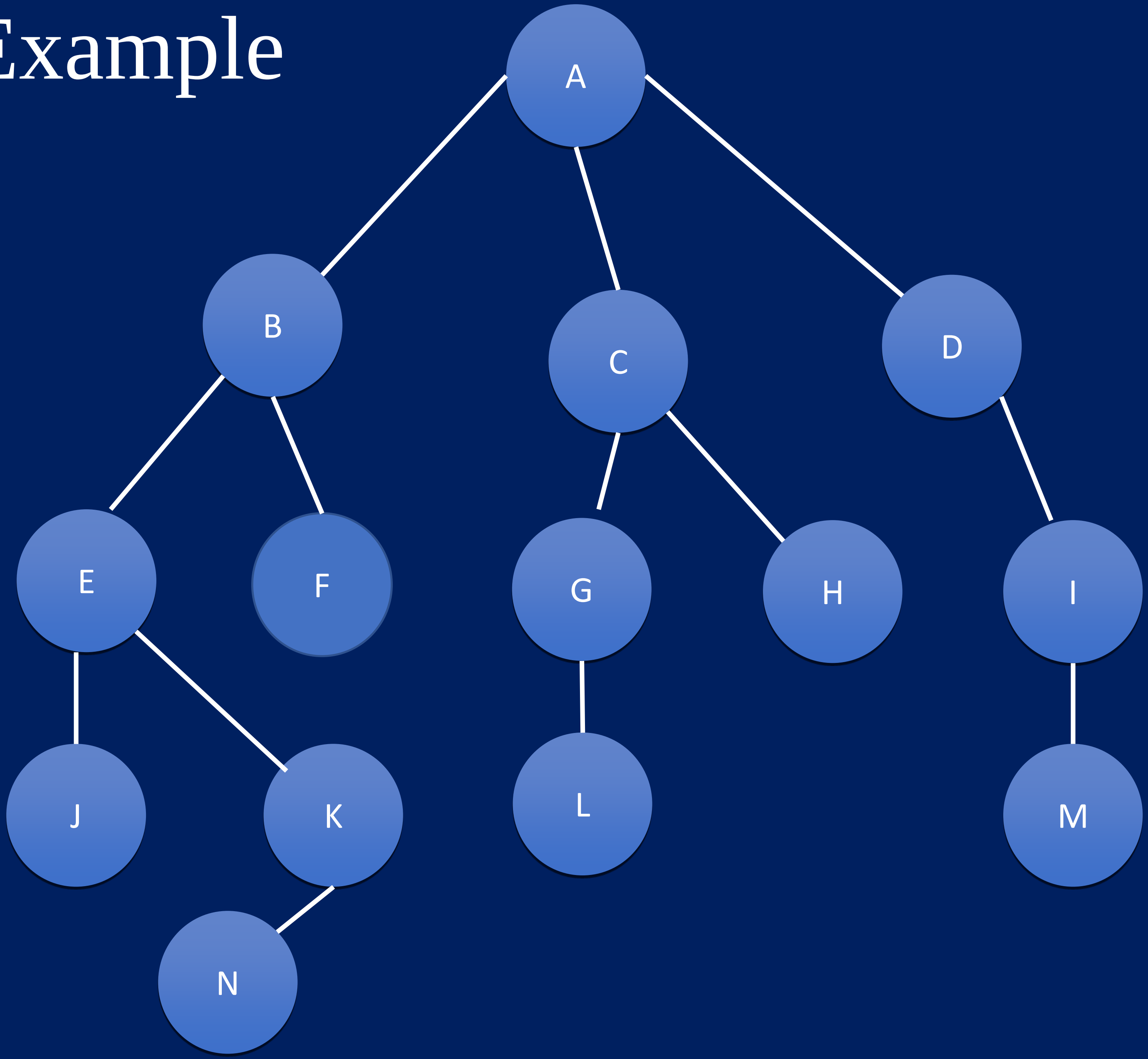
Breadth-first search on a simple binary tree. At each stage, the node to be expanded next is indicated by the triangular marker.

BFS – Adv and Disadv..

- Provides solution if any solution exist.
- For more than one solution, it provides min solution which requires least steps.

- Memory exhausted
- Requires lot of time to find the solution.

Example



BFS

Optimal	YES
Time Complexity	$O(b^d)$

b is the branching factor
d is the depth or levels

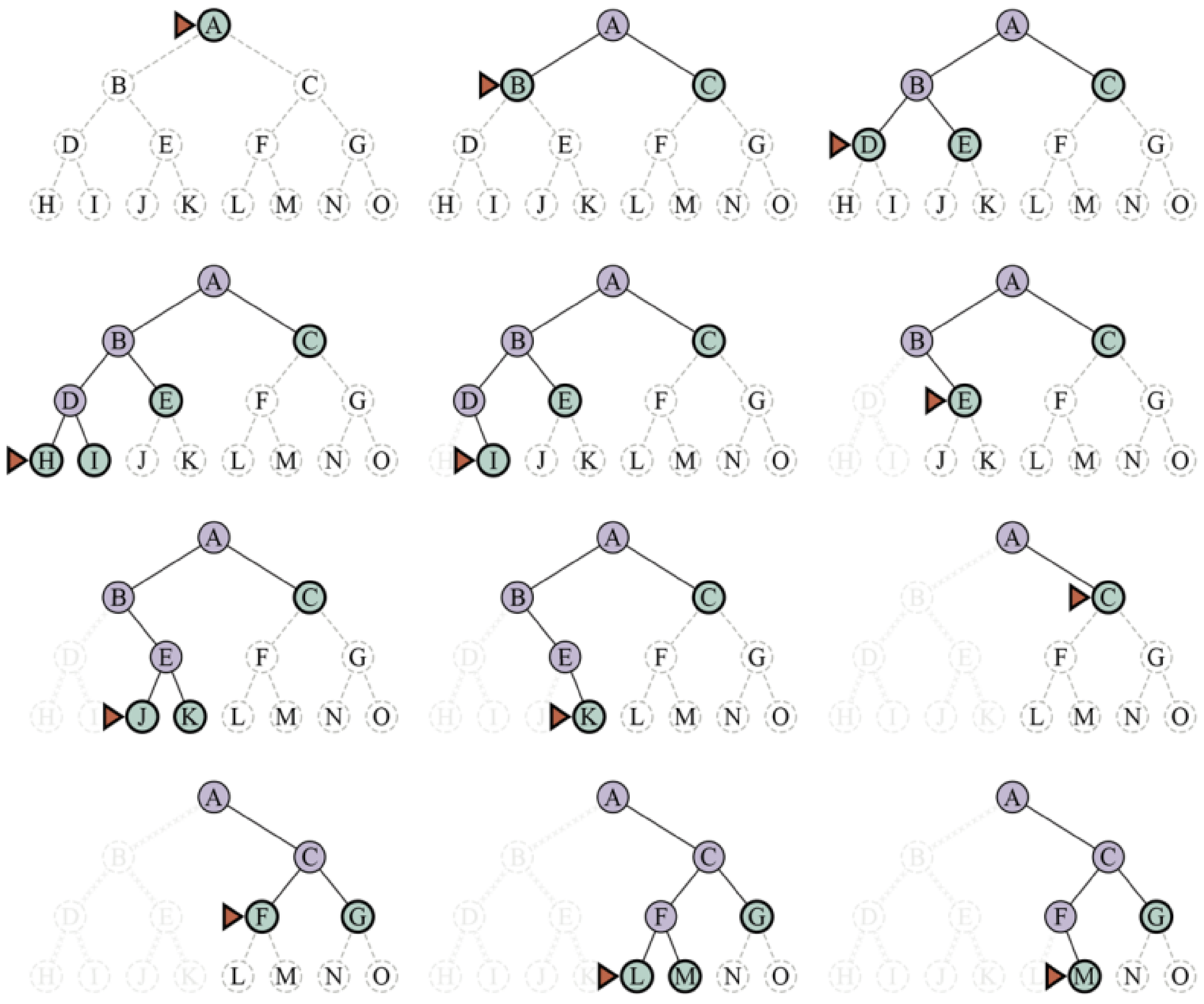
Depth-first search

Depth-first search always expands the deepest node in the frontier first.

The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. The search then “backs up” to the next deepest node that still has unexpanded successors.

Depth-first search is not **cost-optimal**; it returns the first solution it finds, even if it is not cheapest.

Stack – LIFO


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

Example

DFS

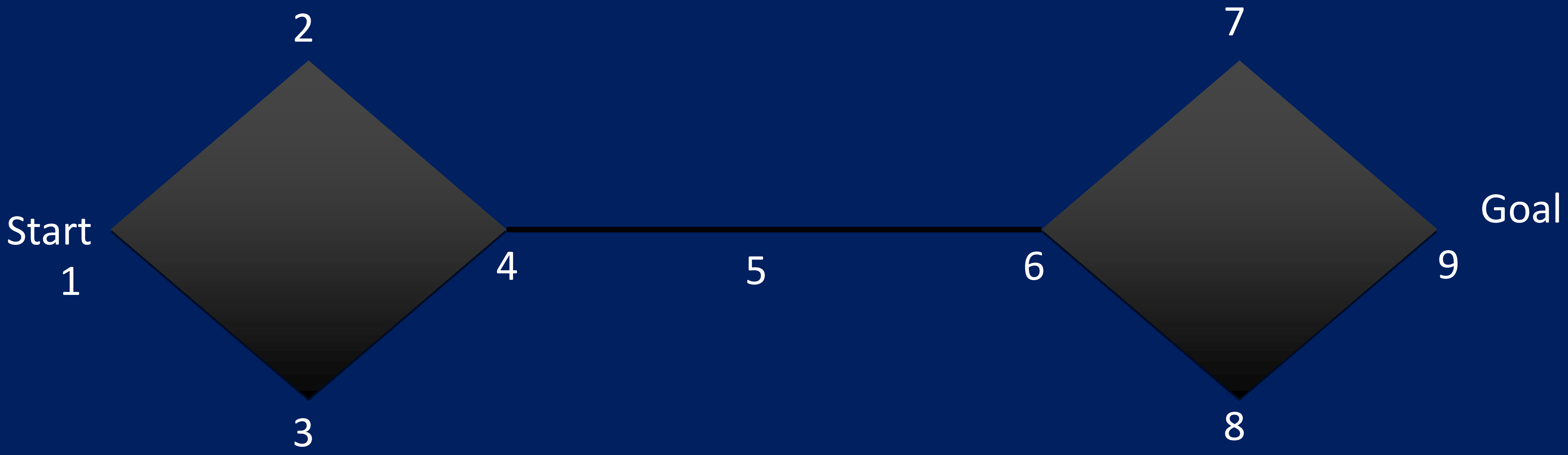
Optimal	No
Time Complexity	$O(b^d)$

b is the branching factor
d is the depth or levels

Bi-Directional Search

- Extension of BFS and DFS
- Two simultaneous search from an initial node to gaol and background from goal to initial. Stop when meet.
- Move in both directions.
- Search stops when there two graphs intersect each other.

Adv:	Disadv:
Fast	*Difficult implementation*
Less memory	*Goal state in advance.*


2
7
Start
1
4
5
6
Goal
9
3
8

Heuristic Approach

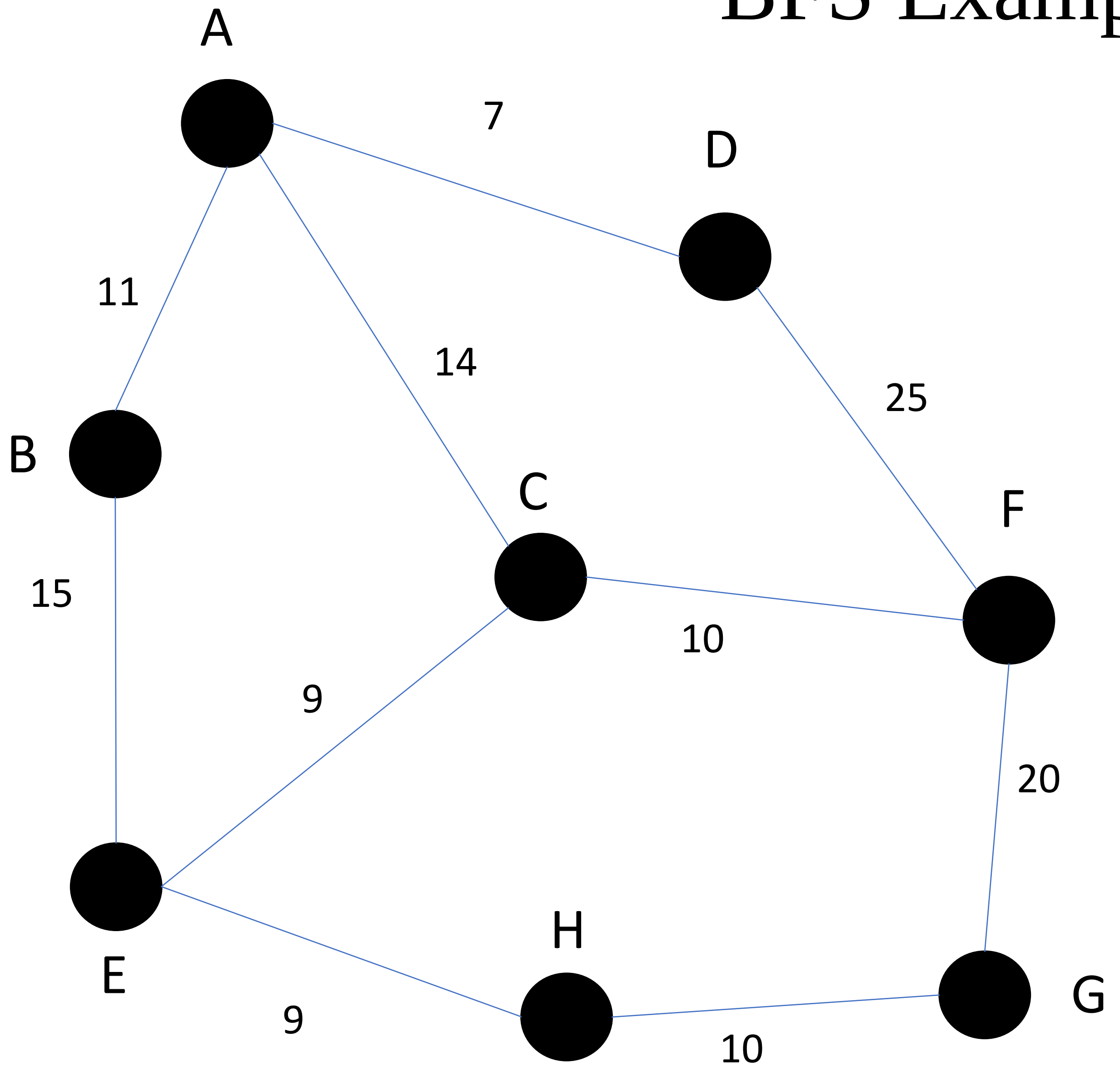
Best First Search

- Select path which appears best at that moment.
- Combination of both DFS and BFS.
- Use Heuristic Function.
- Implemented by Priority Queue.
- Evaluated function $\boldsymbol{f(n) = h(n)}$

Estimated cost of the cheapest path from the state at node n to a gaol state.

BFS Example

BFS Example



h- values

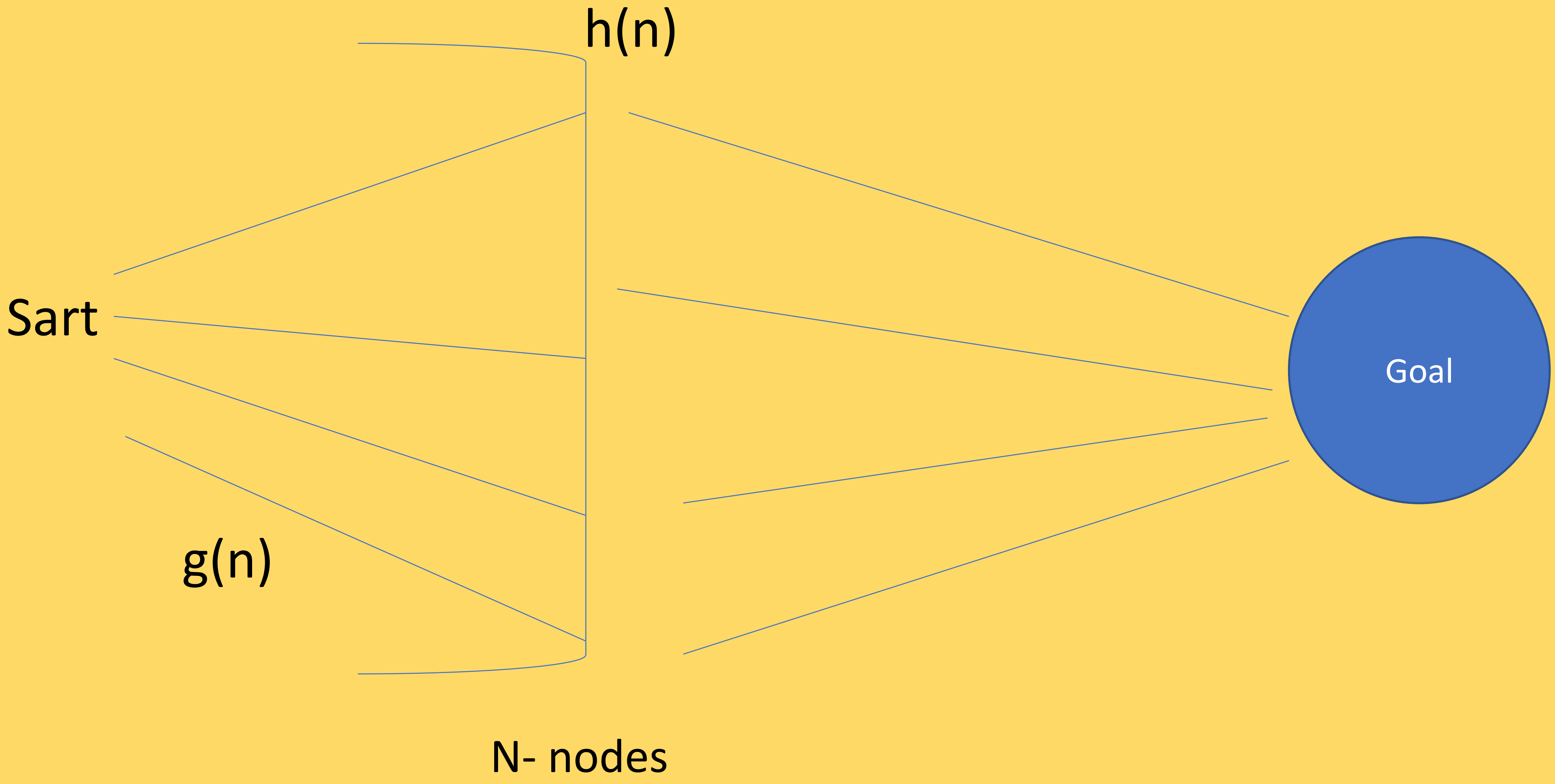
A-G	40
B-G	32
C-G	25
D-G	35
E-G	19
F-G	17
H-G	10
G-G	0

A star Algorithm

- Find the shortest path through the search space using h function.
- It uses h(n) and cost function to reach the node n from start stage.

$$F(n) = g(n) + h(n)$$

- It is similar to UCS except that it uses g(n) + h(n)


h(n)
Sart
g(n)
N- nodes
Goal

Example to be covered.

Still going on….