COOPERATIVE AND ADAPTIVE ALGORITHMS

Otman A. Al-Basir, Spring 2016

COURSE OBJECTIVES

- Motivate the need for algorithms that exhibit a degree of intelligence: logical, computational, and biologically inspired.
- Introduce the concepts of cooperation and adaptations and how they are influencing new methods for solving complex problems.
- Study meta-heuristics, evolutionary computing methods, swarm intelligence, ant-colony algorithms, particle swarm methods.
- Illustrate the use of these algorithms in solving continuous and discrete problems that arise in engineering applications.

CALENDAR DESCRIPTION

- *The course starts by addressing ill-structured problems and the need for computational intelligence methods. It introduces the concepts of heuristics and their use in conjunction with search methods, solving problems using heuristics and metaheuristics, constraints satisfaction.*

- *The course also introduces the concepts of cooperation and adaptations and how they are influencing new methods for solving complex problems. The course starts by illustrating how the concepts of cooperation and adaptation are manifested in nature and how such models are inspiring new types of solution methods.*

CALENDAR DESCRIPTION

Topics to be covered include: search algorithms, game playing, constraints satisfaction, meta-heuristics, evolutionary computing methods, swarm intelligence, ant-colony algorithms, particle swarm methods, adaptive and learning algorithms and the use of these algorithms in solving continuous and discrete problems that arise in engineering applications.

Prereq: Level at least 4A Computer Engineering or Electrical Engineering or Software Engineering.

Antireq: CS 486, SYDE 422/522

COURSE OUTLINE

The following list is an approximate guide for the material to be covered in the course.

- Introduction: goals and definitions of artificial intelligence; intelligent agents and their environment.
- ill-structured problems and need for approximate algorithms, cooperation and adaptation in nature.
- Search: state space problem formulation and representation; uninformed search; heuristic search; iterative improvement; constraint satisfaction; game-playing.
- Review of blind search, use of heuristic search methods and meta-heuristic algorithms
- Trajectory Methods: Tabu search and Simulated Annealing.

COURSE OUTLINE- CONT.

- Genetic algorithms, cooperation in GA.
- Swarm Intelligence: cooperation and adaptation methods inspired by nature.
- Ant Colony algorithms: ACO- cooperative and multi-ant-colonies.
- Particle swarm algorithms: particle swarm optimization, cooperation within the swarms, cooperation among swarms, swarm ensembles.
- Engineering Applications: optimization, routing, text clustering.

WORKLOAD

- **30% Assignment (3)**
- **20% Midterm**
- **50% Final exam**

TEXTBOOKS (OPTIONAL)

- Fundamentals of Computational Swarm Intelligence, Andries P. Engelbrecht, Wiely and Sons, 2006.
- Introduction to evolutionary computing, by Eiben and A, Smith, J.E, Springer, Berlin, 2003.
- Copies of slides will be posted on the course webpage.

REFERENCES PUT ON RESERVE IN THE LIBRARY

- Fundamentals of Computational Swarm Intelligence by Andries P. Engelbrecht, John Wiley and Sons, 2006.
- Swarm Intelligence By James F. Kennedy, Russell C. Eberhart, Yuhui. Shi Published 2001 by Morgan Kaufmann
- Introduction to evolutionary computing, by Eiben and A, Smith, J.E, Springer, Berlin, 2003.
- Modern heuristic techniques for combinatorial problems, by Revees, C., Halsted Press, New York, 1993.

COURSE TIME TABLE AND MATERIAL

- Lecture Time: Mondays, and Fridays 10:00-11:20,
- Lecture Hall: RCH 101,
- Tutorials Tuesdays: TBD.

- Teaching Assistants:

Sepehr Eghbali: sepehr.eghbali@uwaterloo.ca

Pouya Mehrannia: pmehrann@uwaterloo.ca

COURSE MATERIAL

Will be posted on Learn

- Course Outline
- Lecture slides
- Assignments
- Selected References

INTRODUCTION

① AI- Definition and Categorization

② Goals

③ Rationality

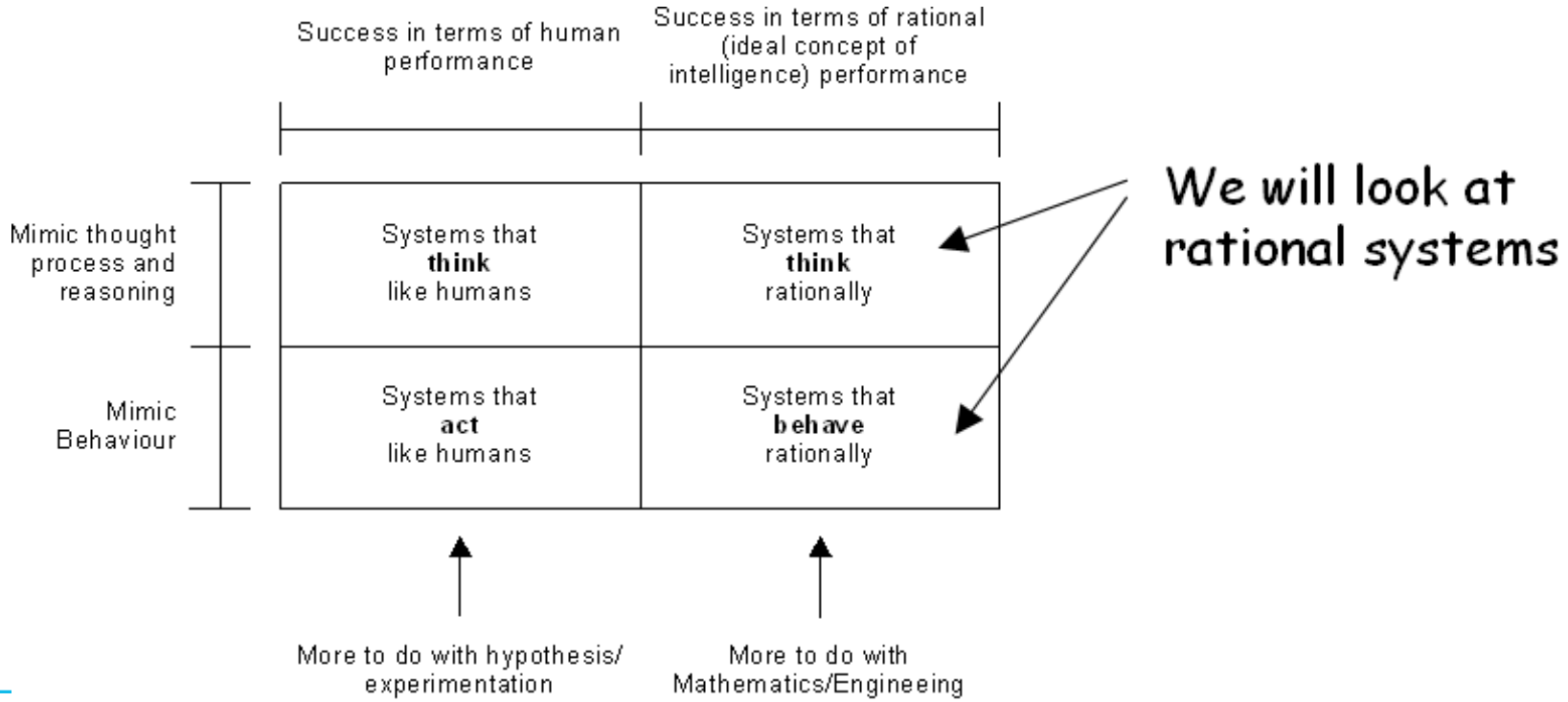
④ Intelligent Agents/Algorithms

⑤ Applications

DEFINITION AND CATEGORIZATION

How should we define Artificial Intelligence (AI) ?

Can organize it into categories based on **goals** and **methodologies.**



AI GOALS

There is a scientific vs. engineering goal for AI.

- Scientific goal for AI is to develop concepts and mechanisms to **understand** biological intelligent behavior.
- Engineering goal for AI is to develop concepts, theory and practice of **building** intelligent machines.

- Viewed differently, our goals might include…
  - Replicate human intelligence and reasoning.
  - Solve hard and knowledge intensive problems.
  - Develop a connection between perceptions and actions.
  - Enhance human-human, human-machine, iteractions.
  - And so forth…

RATIONAL SYSTEMS

Mentioned that in engineering, we are interested in **rational** systems.

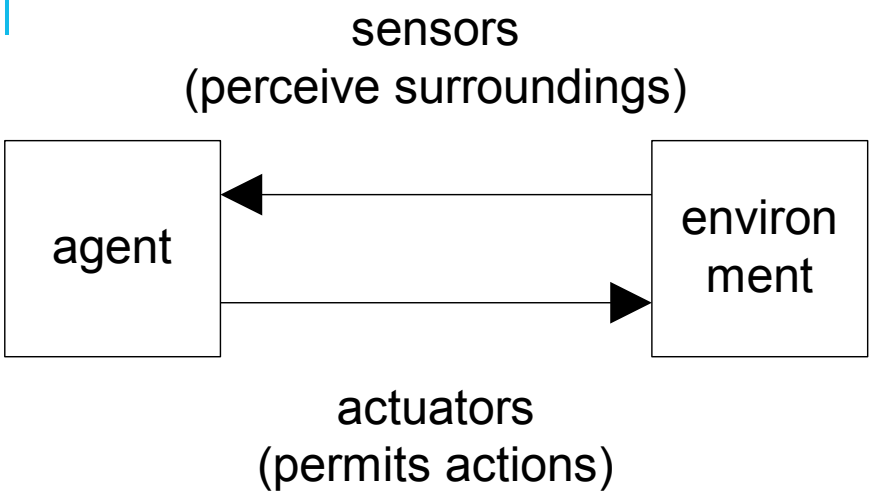
Thinking rationally means using logic to achieve goals via logical inferencing.

- Hard to represent informal knowledge
- Not all problems solvable in this manner (e.g., uncertainty).

Behaving rationally means perceiving the "world" and acting to achieve some goals given a set of beliefs.

- Amenable to computation.
- More general that inferencing (but can use inferencing).
- Actions taken to achieve a goal are not necessarily "correct", but accomplish task at hand.

INTELLIGENT AGENTS/ALGORITHMS



Text introduces the concept of an "agent" and "agent-based systems".

- An agent is something that **senses** its environment and **acts** on it.

We would like to design **rational agents.**

- A **rational agent** is one that for each perceived sequence of events, it does what is expected to maximize performance on the basis of perceptual history and built-in knowledge (leads to concept of **autonomy**).

TYPES OF AGENTS

Different categories of agents (see text):

- **Simple Reflex Agents**
  - Table-lookup approach; needs fully-observable environment.
- **Model-Based Reflex Agents**
  - Adds state information to handle partially observable environments.
- **Goal-Based Agents**
  - Adds concept of goals to augment knowledge to help choose best actions.
- **Utility-Based Agents**
  - Adds utility to decide "good" and "bad" with conflicting goals.
- **Learning Agents**
  - Adds ability to learn situations that affect performance; decides how to change things to improve.

Essentially, agent design is more or less complex depending on the particular characteristics of its **environment**.

ENVIRONMENTS

Agents work within an **environment** with certain characteristics.

It's useful to identify and understand a small number of possible dimensions for an environment that can influence the design of an agent (as previously mentioned).

E.g., is an environment …

- **Fully Observable vs. Partially Observable**
- **Deterministic vs. Stochastic**
- **Episodic vs. Sequential**
- **Static vs. Dynamic**
- **Discrete vs. Continuous**
- **Competitive vs. Co-operative**

POSSIBLE APPLICATIONS OF AI CONCEPTS

- **Game playing:**

  Have well-defined rules with moves that can be searched intelligently.

- **Theorem proving:**

  Proving correctness of this or that (e.g., circuits).

- **Path Planning**

- **Robotics:**

  Intelligent responses to some environment such as obstacle avoidance.

- **Pattern recognition,**

- **Language processing,**

- And so forth…

AI AND ADAPTATION

- To adjust to new or different situations
- Ability to improve behavior
- Evolution
- Learning from instructor, example or by discovery
- Generalization

COOPERATIVE

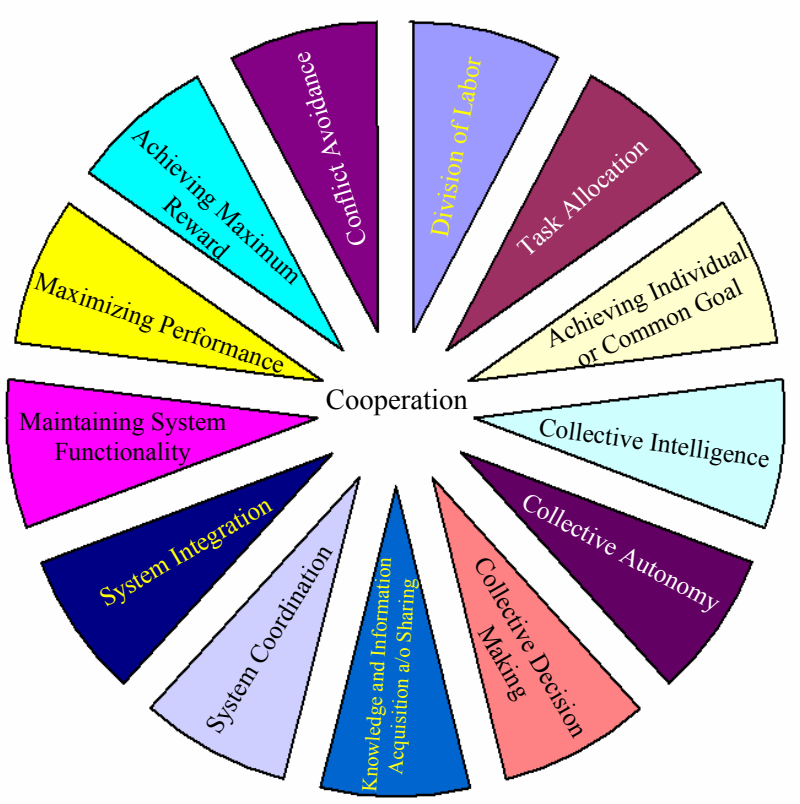
A group to solve a joint problem or perform a common task(s) based on sharing the responsibility for reaching the goal.

- Direct cooperation
- Indirect cooperation
- Independent actions

COOPERATION

Cooperation is the practice of hardware and/or software entities working together in order to achieve certain objective.

- This objective can be, but not limited to,
  - Achieving Individual or Common Goal
  - Division of Labor
  - Collective Autonomy
  - Conflict Avoidance
  - Achieving Maximum Reward
  - Maintaining System Functionality
  - Knowledge and Information Acquisition and/or Sharir
  - Achieving Collective Intelligence



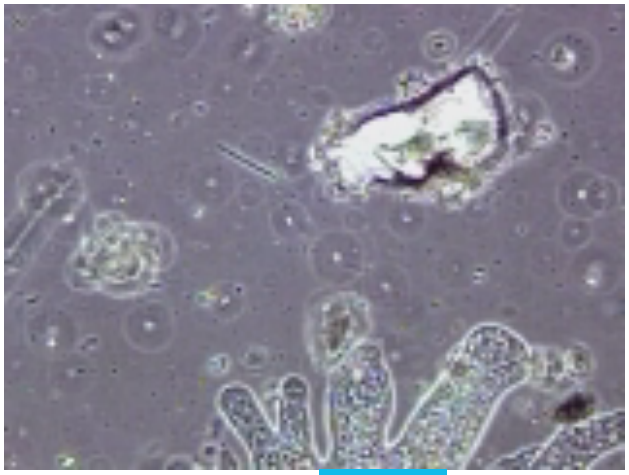
INSPIRATION FROM NATURE

- Cells, immune system, brain cells and DNA
- Ant Colonies, Bee Hives
- Swarms of Birds, fish
- Animals: lions, monkeys,..
- Human: collective behavior

Amoeba or ameba are one-celled organisms belonging to the phylum Sarcodina of the kingdom Protista.

Slime Mold

Amoeba engulfing prey



Slime mold is a kind of colony of unicellular amoeba. It acts like a distributed system consisting of numbers of self-induced oscillators.

Searching Behavior of Slime Mold

When food-stuffs in the environment are exhausted, amoebae begin to aggregate, after which amoebae form multi-celled slugs.

DNA

DNA carries generic instructions for the biological development of all cellular forms of life and many viruses.

It is sometimes referred to as the molecule of heredity as it is inherited and used to propagate traits.

During reproduction, it is replicated and transmitted to offspring

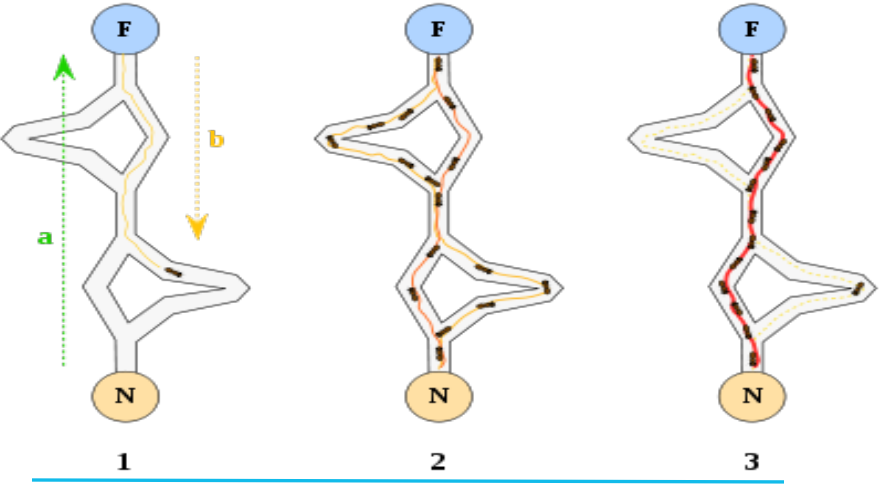
ANTS AND BEES

Ants: about 10^{15} on earth

The total weight of ants is about the total weight of humans

Bees cooperate in building hives

Division of labor



Ants' agricultural behaviour

Bird Flocking

Staying together but not colliding

Bird flocking is a behavior controlled by three simple rules:

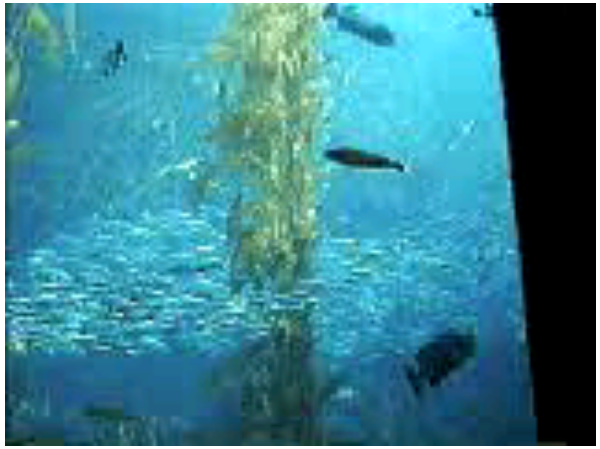
Separation is the behavior that causes an agent to steer away from all of its neighbors.

Cohesion - steer towards average position of neighbors

Alignment - steer towards average heading of neighbors

Fish Schooling

The term schooling describes the behavior of a school of fish of similar size and body orientation, generally cruising in the same direction



Cruising in a close group has advantages in the energy consumption, one fish utilizing the pressure field created by the next fish.

Clip Source: Marc Kummel http://www.rain.org/~mkummel/stumpers/20sep02a.html

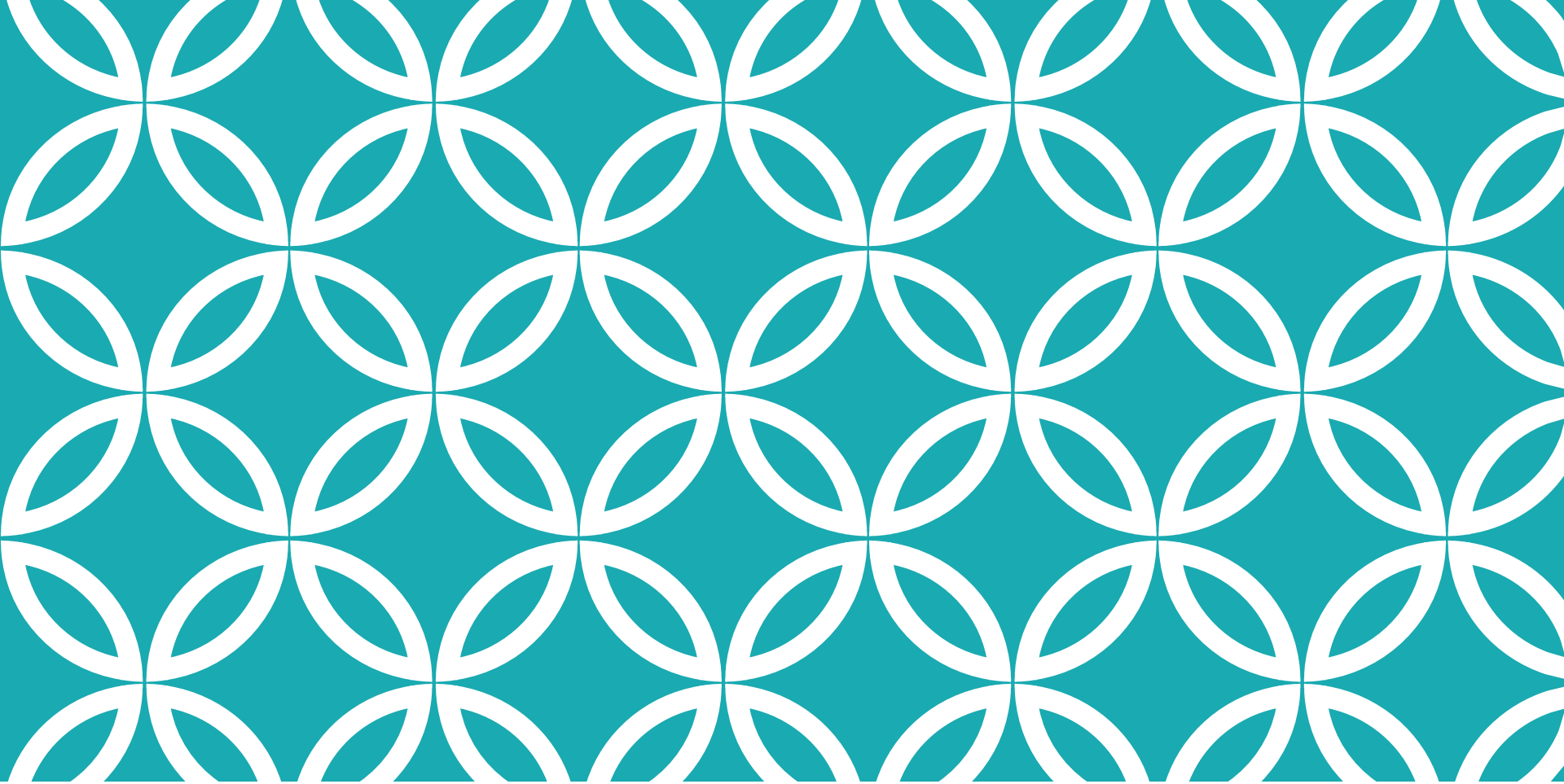
HUMAN

Games: cooperation in playing Self organizing patterns in traffic

Emergent behavior from investors affecting the market in their own way.

BEHAVIOR

Behavior	Inspired Models/Algorithms	Applications
Bacterial Intelligence	Bacterial Chemotaxis Model, EC	Neural networks, robotics configurations
Chromosome behavior	Genetic Algorithm, EC	Optimization Problems in Distributed Computer Systems, Cooperative Manipulations
Ants	Ant Colony Optimization (ACO)/ AC clustering	Machine scheduling, Network Routing, Allocation and assignments, text classification and clustering, graph problems
Swarms	Particle Swarm Optimization (PSO)	Optimization problems, scheduling, mobile robots and control, neural network training,



Classes of Problems

WELL/ILL STRUCTURED PROBLEMS

WELL STRUCTURED PROBLEMS

Problem for which the existing state and desired state are clearly identified, and the methods to reach the desired state are fairly obvious. See also ill structured problem.

WELL/ILL-STRUCTURED PROBLEMS

ILL STRUCTURED PROBLEMS

Situation in which its existing state and the desired state are unclear and, hence, methods of reaching the desired state cannot be found. See also well structured problem.

SOLUTION STRATEGIES

Continuation of week 1
Lecture

SOLUTION STRATEGIES FOR ILL-STRUCTURED PROBLEMS

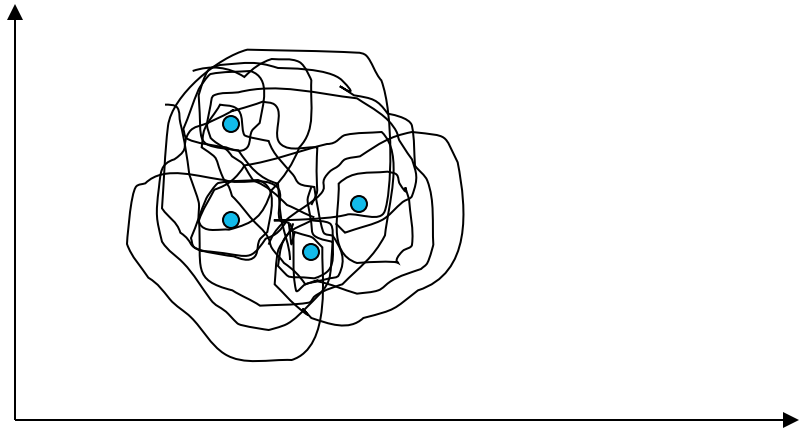
- Retrieving a solution or an answer.
- Starting from a guess and then improve on it .
- Searching among alternatives.
- Searching forward from the problem to the answer.
- Searching backward from a goal to the situations of the problem

EXAMPLE OF WELL-STRUCTURED PROBLEM BUT PRACTICALLY ILL-STRUCTURED

THE CLUSTERING PROBLEM

Given *n* objects, group them in c groups (Clusters) in such a way that all objects in a single group have a “natural" relation to one another, and objects not in the same group are somehow different

MANY WAYS TO CLUSTER OBJECTS



SET PARTITIONING

- Number of possible partitions is given by

$$N(n,c) = \left(\frac{1}{c!}\right)\sum_{m=1}^{c}(-1)^{c-m}\binom{c}{m}m^n$$

For example if *n=50 and c=4,* the number is $5.3x10^{28}$. If *n* is increased to 100, the number becomes $6.7x10^{58}$.

So enumerating all possible partitions for large problems is practically not feasible

OPTIMIZATION PROBLEMS

Optimization algorithms

- Exact algorithms
  - Find the optimal solution,
  - High computational costs.
- Approximate algorithms
  - Find sub-optimal solutions,
  - Low computational costs.

OPTIMIZATION ALGORITHMS

Approximate algorithms can be divided to

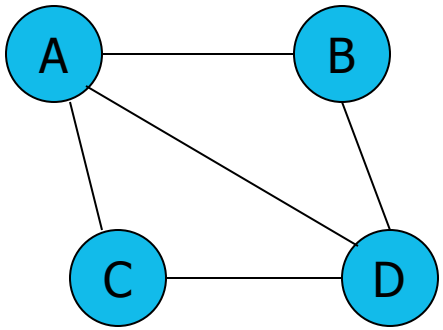
- Constructive methods

  start from scratch and try to build the complete solution by adding one component at a time

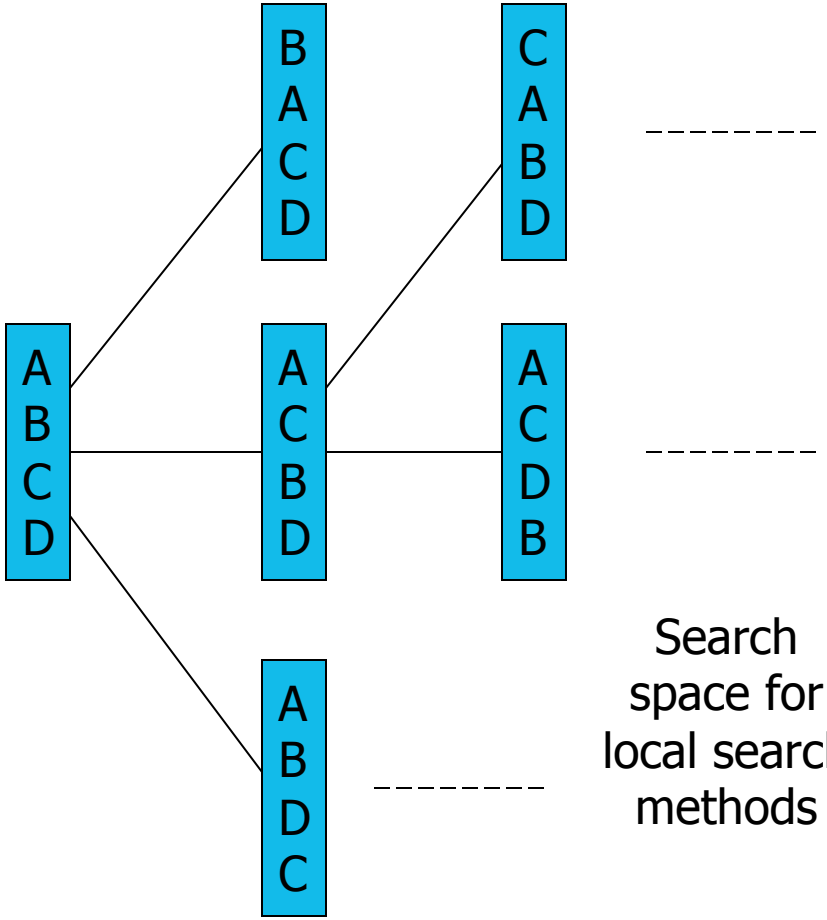
- Local search methods

  start from an initial solution and iteratively tries to replace the current solution with better one

OPTIMIZATION SEARCH SPACE



Search space for constructive methods



Search space for local search methods

SEARCH AND META-HEURISTIC METHODS

Introduction

SEARCH

Can often solve a problem using **search.**

Two requirements to use search:

Goal Formulation.

- Need goals to **limit search** and **allow termination.**

Problem formulation.

- Compact representation of problem space (**states**).
- Define **actions** valid for a given state.
- Define **cost** of actions.

Search then involves moving from state-to-state in the problem space to find a goal (or to terminate without finding a goal).

SEARCH
Central in many AI systems
Theorem proving, VLSI layout, game playing, navigation, scheduling, etc.

REQUIREMENTS FOR SEARCHING

Define the problem

- Represent the search space by states
- Define the actions the agent can perform and their cost

Define a goal

- What is the agent searching for?

Define the solution

- The goal itself?
- The path (i.e. sequence of actions) to get to the goal?

ASSUMPTIONS

Goal-based Search

Environment

- Fully observable
- Deterministic
- Sequential
- Static
- Discrete

PROBLEM FORMULATION

Search requires a well-defined problem space including:

- **Initial state**
  - A search starts from here.
- **Goal state and goal test**
  - A search terminates here.
- **Sets of actions**
  - This allows movement between states (successor function).
- **Concept of cost** (action and path cost)
  - This allows costing a solution.

The above defines a **well-defined state-space formulation** of a problem.

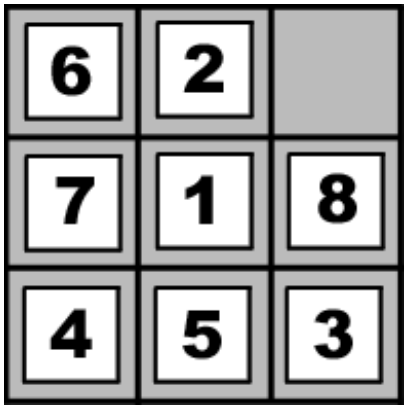
Note: A state can represent either a complete configuration of a problem (e.g., 8-puzzle) or a partial configuration of a problem (e.g., routing).

WELL-DEFINED PROBLEM EXAMPLE

Initial state

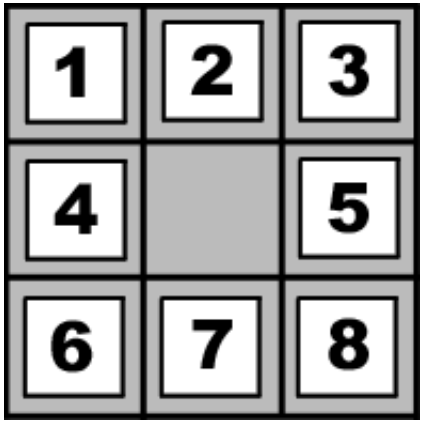
Action

- Move blank left, right, up or down, provided it does not get out of the game



Goal test

- Are the tiles in the "goal state" order?



Cost

- Each move costs 1
- Path cost is the sum of moves

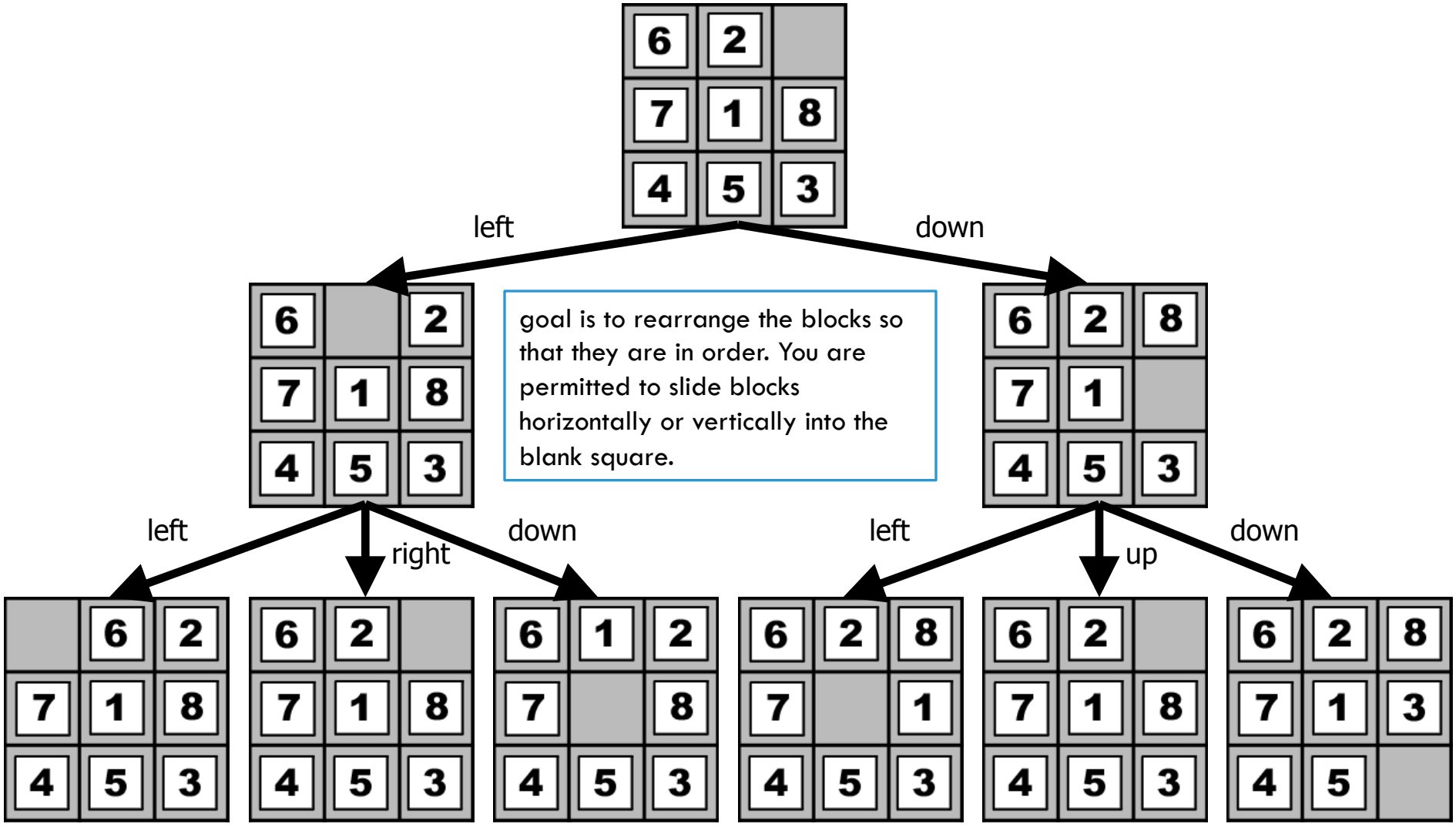
PROBLEM FORMULATION EXAMPLE – 8 PUZZLE

Want to take an initial arrangement of tiles and get them into a desired arrangement.

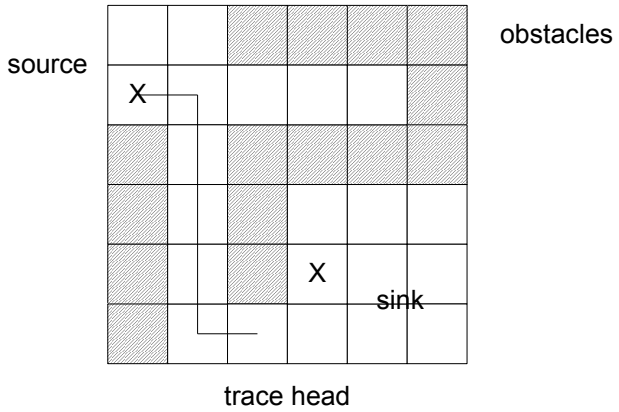
- States: encode location of each of 8 tiles and the blank.
- Initial State: any arrangement of tiles.
- Goal State: predefined arrangement of tiles.
- Actions: slide blank UP, DOWN, LEFT or RIGHT (without moving off the grid).
- Goal Test: position of tiles and blank match goal state.
- Cost: sliding the blank is 1 move (cost of 1). Path cost will equal the number of moves from initial state to goal state.

Solution is a path of moves to get to the goal state. Fewest moves is best.

EXAMPLE: 8-PUZZLE



PROBLEM FORMULATION EXAMPLE – ROUTE FINDING



Problem is to connect a source to a sink while avoiding obstacles on 2-D grid.

- States: ordered pair (x,y) of the trace head.
- Initial State: trace at location of source.
- Goal State: trace at location of sink.
- Actions: move trace head UP, DOWN, LEFT or RIGHT (without moving off the grid and avoiding obstacles).
- Goal Test: trace at location of sink.
- Cost: moving trace head costs 1. Path cost is length of trace.

Solution might be (i) trace with shortest path or (ii) a path if one even exists.

WELL-DEFINED PROBLEM EXAMPLE

Travelling salesman problem

- Find the shortest round trip to visit each city exactly once

Initial state

- Any city

Set of actions

- Move to an unvisited city

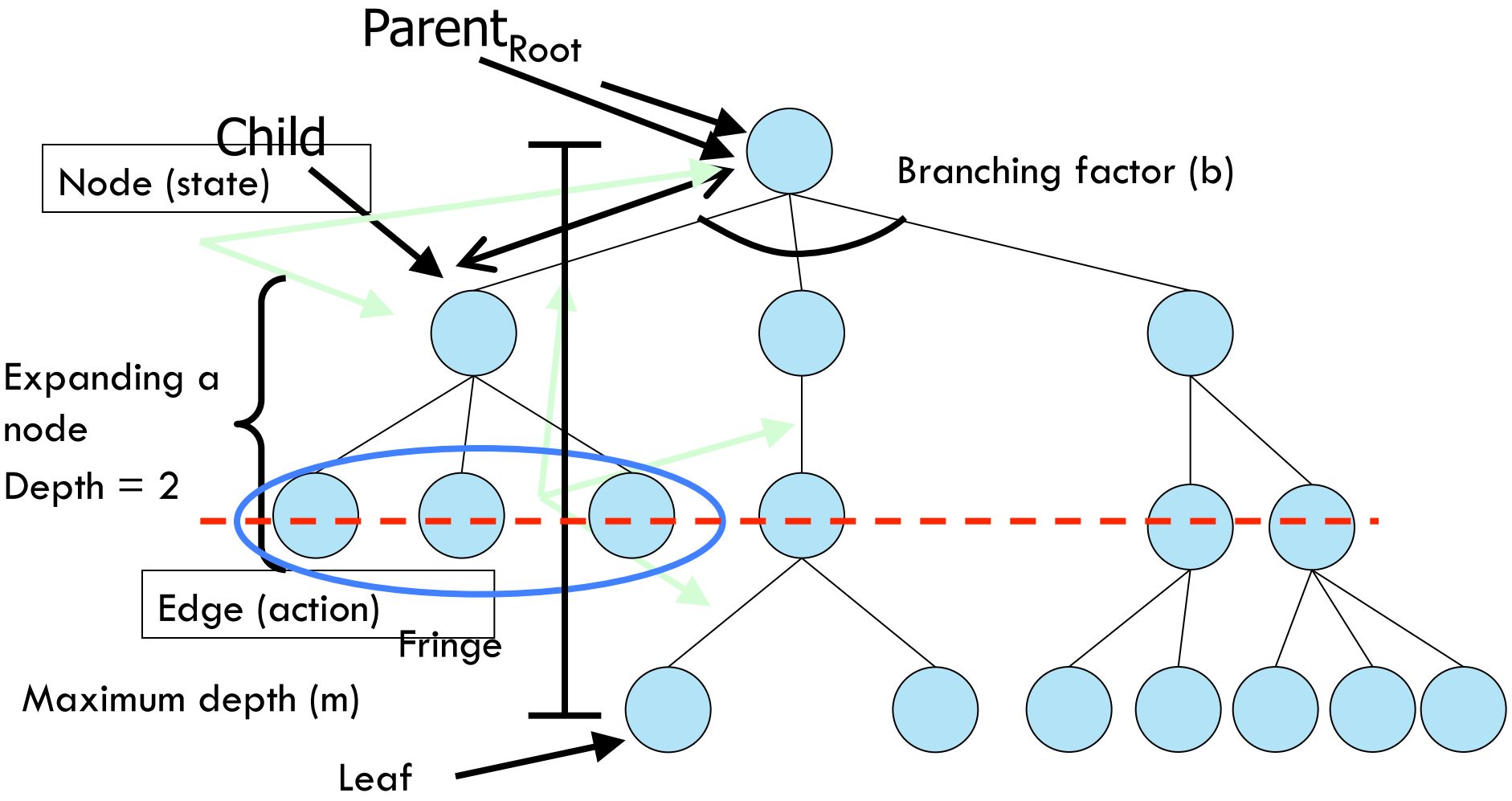
Goal test

- Is the agent in the initial city after having visited every city?

Concept of cost

- Action cost: distance between cities
- Path cost: total distance travelled

SEARCH TREE


Parent
Root
Child
Node (state)
Branching factor (b)
Expanding a
node
Depth = 2
Edge (action)
Fringe
Maximum depth (m)
Leaf

PROPERTIES OF SEARCH ALGOS.

Completeness

- Is the algorithm guaranteed to find *a* goal node, if one exists?

Optimality

- Is the algorithm guaranteed to find *the best* goal node, i.e. the one with the cheapest path cost?

Time complexity

- How many nodes are generated?

Space complexity

- What's the maximum number of nodes stored in memory?

COMMENTS ON SEARCH TREES

Search trees are superimposed over top of the graph representation of a problem.

While the graph might be finite, the search tree can be either finite or infinite.

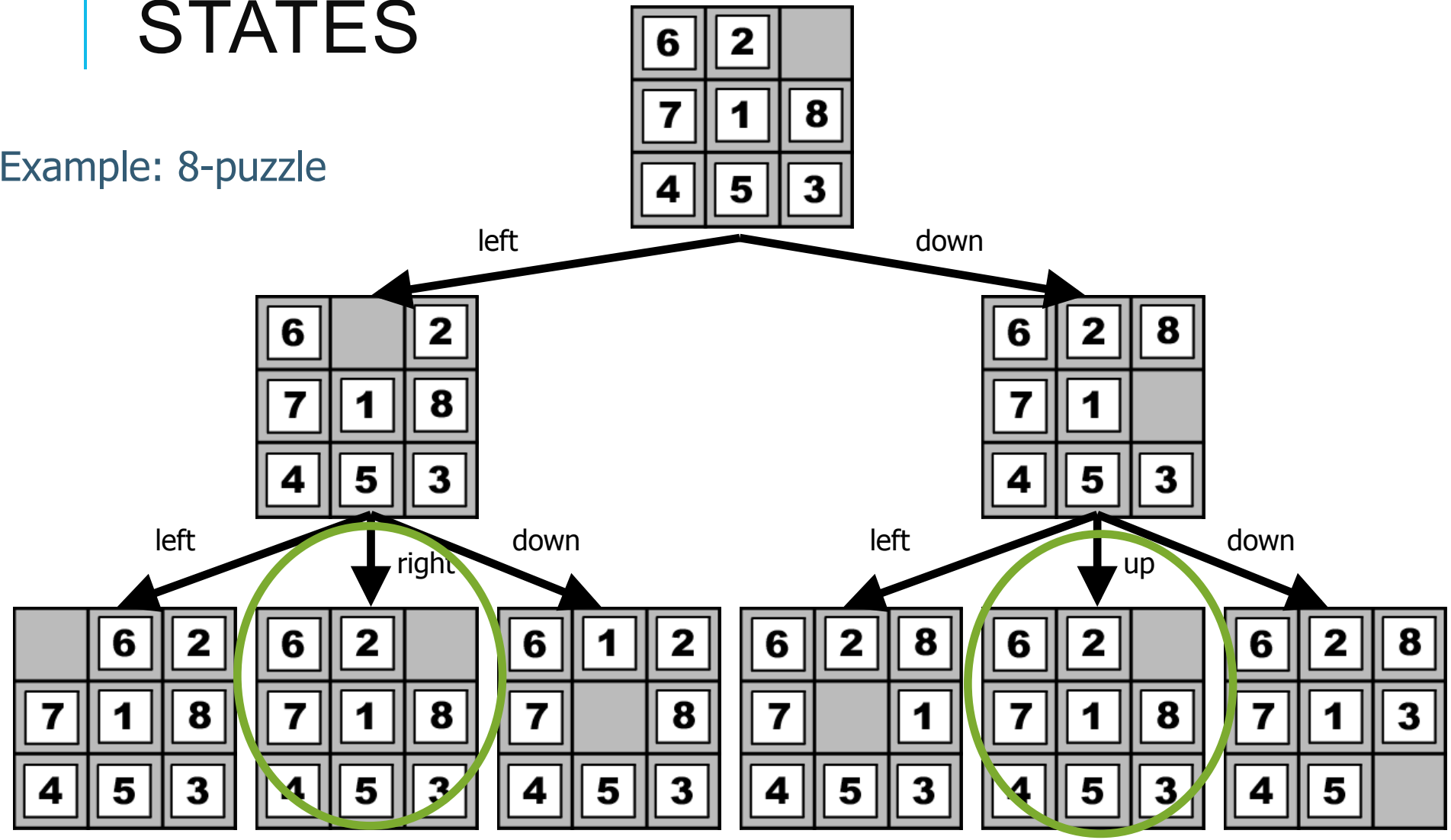
- Infinite if we allow repeated states due to reversible actions and/or cycles of actions.

Some useful terminology:

- The maximum number of children possible for a node, b, in a search tree is called the **branching factor**.
- A finite tree has a **maximum depth**, d.
- Any node in the search tree occurs at a **level**, l, in the tree ($l \cdot d$).

REPEATED STATES

Example: 8-puzzle



TEMPLATE OF GENERIC SEARCH

Given concept of graph and search tree, generic search is a repetition of **choose, test, and expand**.

A particular search strategy (uninformed or informed, more in a minute …) influences how we choose the next node to consider in the search! (this is where types of searches differ).

- Generally use a **queue** to store nodes on the fringe to be expanded. Different search strategies use different queue structures.

META-HEURISTICS

Meta-heuristics are algorithms that combine heuristics in a higher level framework aimed at efficiently and effectively exploring the search space,

A metaheuristic search is a non-exhaustive search algorithm with internal heuristic.

Metaheuristic searches are therefore often hybrid searches where the:

first search identifies neighborhoods that might be valid locations for the search term, and

the second search intensifies the search in order to find the exact right answer.

There is only one problem with this search technique, and that is that it might skip over the right answer, simply because it wasn't in any of the neighborhoods it searched.

Type:

- Trajectory methods, which handles one solution at a time,
- Population-based, which handles multiple solutions.

TRAJECTORY METHODS

Variants of local search methods to avoid getting stuck at local optimum.

Tabu Search

- Among the most cited meta-heuristics used for optimization,
- It uses memory structures to:
  - Escape local minima,
  - Implement an explorative strategy. Avoid revisiting visited nodes.

TRAJECTORY METHODS

Simulated Annealing

- Mimics the physical annealing process
- Basic idea is to escape local minima by allowing some "bad" moves.
- Used for locating a good approximation of the global optimum in large search spaces

POPULATION BASED: EVOLUTIONARY COMPUTING

Genetic Algorithm:

Mimics the biological genetic process

Starts with an initial solution and change it using mutation and crossover type operations to produce better solutions

POPULATION BASED: SWARM INTELLIGENCE

Inspired from the collective behavior of social insect, flocks of birds or fish schools.

Complex tasks can be carried out through cooperation.

ACO AND PSO

Ant Colony Optimization (ACO):

- Inspired by the foraging behaviour of ants.

Particle Swarm Optimization (PSO):

- Originally developed to simulate the behaviour of a flock of birds or school of fish looking for food.