ARTIFICIAL INTELLIGENCE AND NEURAL NETWORKS

Lecture 5a – Search, Heuristics, and Intelligent Problem Solving

Chizhi Chris ZHANG

zhangchizhi@ciomp.ac.cn

Advanced Computing and Digital Technology Research Center

University of Chinese Academy of Sciences

Spring 2026

Today's Question

What we are trying to answer

How do we make good decisions when a problem has too many possible answers to check one by one?

Why this belongs in AI

Not every AI problem is "learn a function from data." Some problems ask us to arrange, route, schedule, or allocate under limits.

What changes from AI4

AI4 was mainly about predicting from data with trees and ensembles. AI5 moves to problems where the hard part is searching for a good plan.

From AI4 to AI5

Last time

We studied models that learn patterns from rows of data and then make predictions for new cases.

Today

We shift to problems where the answer is not a label or a number, but a choice among many possible arrangements.

One sentence

Machine learning often asks, “what will happen?” Search often asks, “which plan should we try?”

Why Search Problems Matter

These problems are everywhere

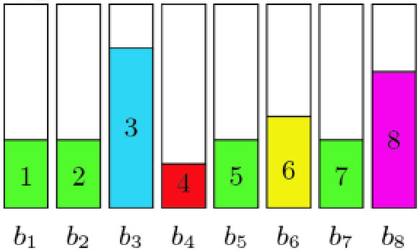
Course timetables, warehouse layouts, delivery routes, hospital appointments, exam seating, and robot navigation all ask for a good arrangement under constraints.

Why they feel different

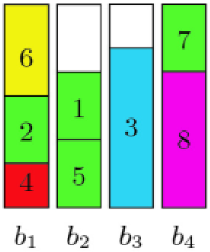
There may be many legal answers, many bad answers, and no cheap way to prove the best one by brute force.

This is why search feels practical. The system is not waiting for a beautiful theory. It is waiting for a workable decision.

A Running Example



A feasible solution, with 8 bins



An optimal solution, with 4 bins

The task

Pack items into bins without breaking capacity limits, while using as few bins as possible.

Why this example works

The goal is easy to understand, but the number of possible packings grows quickly. That makes it a clean story for search.

Why Brute Force Fails

The core difficulty

If every new item can be placed in several different ways, the number of candidate solutions explodes much faster than our intuition expects.

What this means in practice

A method that feels fine on a toy case may become useless when the real problem has hundreds or thousands of decisions.

The key point

Intelligent search is mostly about learning where not to spend time.

Model Before You Search

A good search model answers four questions

- What counts as one state?
- What moves are allowed from that state?
- What makes a state legal or illegal?
- How do we judge whether one state is better than another?

A campus example

If we build an exam timetable, a “state” may be a partial schedule, an “action” may be placing one exam, and a “bad move” may create a clash for students.

Constraints Shape the Whole Problem

Constraints are not decoration

They decide which parts of the search space are even worth entering. A clever method that keeps breaking the rules is still a bad method.

A common mistake

Students sometimes optimize the objective first and remember the constraints later. In real systems, that usually produces elegant nonsense.

Good search starts with careful problem definition, not with a favorite algorithm.

Greedy as the First Baseline

Greedy idea

At each step, choose the move that looks best right now according to a simple local rule.

Why people like greedy methods

They are easy to explain, fast to run, and often good enough to give a usable starting answer.

What greedy gives up

It commits early without seeing the whole future.

A Small Greedy Story

Imagine packing books into boxes

One simple rule says: take the next book and place it into the box with the least leftover space that still fits it.

Why this feels sensible

The rule tries not to waste room, and it makes each decision quickly without rethinking the whole packing plan.

This is the appeal of greedy search: a short rule can turn a messy problem into a sequence of immediate decisions.

When Greedy Helps

Good situations for greedy thinking

Greedy works best when local clues are strong, time is short, and a decent answer quickly is more valuable than a perfect answer slowly.

A practical example

If a delivery company must make a route plan in seconds, a strong baseline may be more useful than a theoretically better method that finishes too late.

When Greedy Misleads

The weakness

A move that looks smart now can block a much better arrangement later.

The warning

“Fast” and “reasonable” are not the same as “globally wise.”

Why this happens

Search problems often contain delayed consequences. The cost of a decision may only appear several steps later.

Baseline, Not Belief

How good teams use greedy methods

They treat greedy as a first comparison, not as a religion. A simple baseline helps us see whether a more complex method is really earning its extra cost.

Why this matters in class

If a fancy method barely beats a greedy baseline, then the fancy method may not be worth the trouble.

Search Means Moving Through States

Think of it this way

Think of search as walking through a large space of partial solutions. Each move changes the current state into a nearby one.

Why this picture helps

Once we think in states and moves, ideas like neighborhood, restart, and heuristic evaluation become easier to understand.

The algorithm is not magic. It is deciding where to step next and when to stop.

Local Search Intuition

Local search idea

Instead of constructing the whole answer from scratch, start with one complete candidate and keep improving it by small edits.

A familiar analogy

If a timetable is messy, we often do not throw it away. We swap one exam, move one room, or change one slot at a time.

Hill Climbing in Plain Language

The rule

Look at nearby alternatives and move to one that improves the score. Repeat until no nearby move helps.

Why it is attractive

It is simple, cheap, and often much better than random guessing.

Hill climbing does not promise the best answer. It promises disciplined improvement from where you are.

Why Local Optima Trap Us

The trap

You can reach a state that is better than all nearby states, even though a much better region exists somewhere else.

A mountain metaphor

If you only walk uphill, you may stop on a small hill and never discover the taller mountain beyond the valley.

Important correction

Stopping locally is not the same as solving globally.

How Restart and Randomness Help

Two common tricks

Start from several different initial points, or allow occasional random moves so the search can escape a bad basin.

Why this is not cheating

Randomness is often a practical tool for avoiding predictable traps, not a sign that the method is weak.

Anytime Methods Fit Real Decisions

What “anytime” means

The algorithm can return its best answer so far if time runs out, and often improves that answer if given more time.

Why operations people care

In many real settings, a decent answer now is more useful than a perfect answer after the decision window has closed.

This is one reason heuristics remain relevant even when exact optimization exists in theory.

Heuristics Give Search a Compass

What a heuristic is

A heuristic is a rule of thumb that estimates which choices seem more promising, so the search spends effort more selectively.

The plain-language version

It is not a proof. It is guidance.

A-Star in One Formula

The scoring idea

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

How to read it

$g(n)$ is the cost we have already paid to reach state n. $h(n)$ is our estimate of the remaining cost. The search follows states with the most promising total score.

The formula matters less than the story: part known, part estimated, both used together.

What Admissible Really Means

One useful condition

$$h(n) \leq h^*(n)$$

The heuristic never overestimates the true remaining cost.

Why this matters

If the estimate is optimistic in this careful way, A* keeps a path to the optimal solution open.

A common misunderstanding

Admissible does not mean “accurate.” It means “safely optimistic.”

Heuristic Quality Changes Everything

A weak heuristic

If the estimate tells us little, search behaves closer to blind exploration and wastes time on many unhelpful states.

A strong heuristic

If the estimate captures the structure of the problem well, search can focus on a much smaller and more useful region.

The search algorithm and the heuristic should be thought of as a pair.

Speed and Optimality Pull on Each Other

The tension

If we demand guaranteed optimality, we often spend more time and memory. If we relax that demand, we can move faster.

The real question

Do we need the best answer, or do we need a strong answer before the situation changes?

Designing a Heuristic Is a Modeling Skill

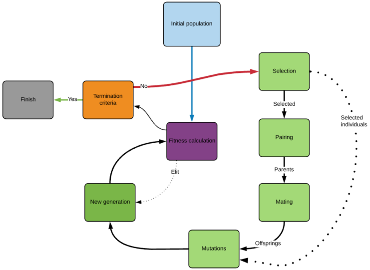
Where good heuristics come from

They usually come from domain insight: geography for route planning, capacity logic for packing, conflict structure for scheduling, or physics for robot movement.

What makes this interesting

The same search algorithm can look brilliant in one domain and weak in another because the heuristic knowledge changed.

Genetic Algorithms Tell a Different Story



The rough idea

Keep a population of candidate solutions, score them, keep better ones, and generate new candidates by recombination and mutation.

Why people remember it

It feels different from step-by-step local improvement because it searches through many candidates at once.

What Crossover and Mutation Are Trying to Do

Crossover

Borrow useful pieces from different candidate solutions and see whether the combination is stronger.

Mutation

Introduce small random changes so the search does not become too uniform too early.

The language sounds biological, but the real purpose is practical: explore, combine, and avoid stagnation.

Why Genetic Algorithms Can Help

Where they are attractive

They can be useful when the landscape is rough, the solution is naturally encoded as a sequence or structure, and local moves alone keep getting stuck.

A simple intuition

Instead of betting on one path, GA keeps several possibilities alive and lets them compete.

Where Genetic Algorithms Disappoint

The hard truth

GA performance depends heavily on representation, fitness design, selection pressure, mutation rate, and stopping rules.

Why this matters

If those choices are poor, the method can burn a lot of computation without producing a satisfying answer.

The correction to the myth

"Inspired by nature" does not mean "automatically powerful."

Hybrid Strategies Usually Win

What happens in real projects

Teams often mix methods: a greedy start, a heuristic search phase, and then local improvement or repair.

Why hybrids make sense

Different stages of a problem reward different kinds of search behavior.

How Teams Actually Choose Methods

The practical checklist

How much time do we have? How large is the search space? Do we need proof of optimality? How costly is a bad answer?

What this means

Method choice is rarely about elegance alone. It is about risk, resources, and decision timing.

Evaluation Should Match Operations

What to evaluate

Not only answer quality, but also runtime, memory use, robustness, and how gracefully the method behaves when the problem size grows.

A classroom warning

An algorithm that looks excellent on one benchmark may fail badly when the constraints, scale, or deadlines change.

Good evaluation asks, “good for which environment?”

Why AI5 Leads to NN5

What AI5 adds to the course

We have now seen an AI route that solves problems by searching through possibilities rather than by fitting one predictive model from data.

Why NN5 is next

NN5 returns to neural models, but with a new lesson: when data has visual structure, the network architecture must respect that structure instead of flattening it away.

Summary

- Search problems ask us to choose among many possible arrangements under constraints.
- Greedy methods are useful baselines because they are simple and fast, but they can be short-sighted.
- Local search improves a current candidate step by step, yet can get trapped in local optima.
- Heuristics guide search by spending computation more selectively rather than blindly.
- Genetic algorithms and hybrid methods remind us that practical problem solving often mixes several ideas.

Next Time

Where this story goes next

NN5 returns to neural networks and asks how vision problems changed the design of the network itself.

What to watch for

Search taught us to respect problem structure. CNNs will make the same point in a different way: good AI design depends on the shape of the data.

Thank You