1. Best-First Search
2. A* and IDA*
3. Local Search Methods
4. Genetic Algorithms
Search procedures differ in the way they determine the next node to expand.

**Uninformed Search:** Rigid procedure with no knowledge of the cost of a given node to the goal.

**Informed Search:** Knowledge of the worth of expanding a node $n$ is given in the form of an *evaluation function* $f(n)$, which assigns a real number to each node. Mostly, $f(n)$ includes as a component a *heuristic function* $h(n)$, which estimates the costs of the cheapest path from $n$ to the goal.

**Best-First Search:** Informed search procedure that expands the node with the “best” $f$-value first.
function TREE-SEARCH(problem) returns a solution, or failure
initialize the frontier using the initial state of problem
loop do
    if the frontier is empty then return failure
choose a leaf node and remove it from the frontier
    if the node contains a goal state then return the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

Best-first search is an instance of the general TREE-SEARCH algorithm in which frontier is a priority queue ordered by an evaluation function $f$. When $f$ is always correct, we do not need to search!
Greedy Search

A possible way to judge the “worth” of a node is to estimate its path-costs to the goal.

\[ h(n) = \text{estimated path-costs from } n \text{ to the goal} \]

The only real restriction is that \( h(n) = 0 \) if \( n \) is a goal.

A best-first search using \( h(n) \) as the evaluation function, i.e., \( f(n) = h(n) \), is called a greedy search.

Example: route-finding problem:
\( h(n) = \text{straight-line distance from } n \text{ to the goal} \)
Heuristics

The evaluation function $h$ in greedy searches is also called a *heuristic* function or simply a *heuristic*.

- The word *heuristic* is derived from the Greek word $\varepsilonυρισκειν$ (note also: $\varepsilonυρηκα$!)
- The mathematician Polya introduced the word in the context of problem solving techniques.
- In AI it has two meanings:
  - Heuristics are fast but in certain situations incomplete methods for problem-solving [Newell, Shaw, Simon 1963] (The greedy search is actually generally incomplete).
  - Heuristics are methods that improve the search in the average-case.

→ In all cases, the heuristic is *problem-specific* and *focuses* the search!
Greedy Search Example
Greedy Search from Arad to Bucharest

(a) The initial state

(b) After expanding Arad

(c) After expanding Sibiu

(d) After expanding Fagaras
a good heuristic might reduce search time drastically
non-optimal
incomplete
graph-search version is complete only in finite spaces
Can we do better?
A*: Minimization of the Estimated Path Costs

A* combines the greedy search with the uniform-search strategy: Always expand node with lowest $f(n)$ first, where

$$g(n) = \text{actual cost from the initial state to } n.$$  
$$h(n) = \text{estimated cost from } n \text{ to the next goal.}$$  
$$f(n) = g(n) + h(n),$$  
the estimated cost of the cheapest solution through $n$.

Let $h^*(n)$ be the actual cost of the optimal path from $n$ to the next goal. $h$ is \textit{admissible} if the following holds for all $n$: 

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

We require that for A*, $h$ is admissible (example: straight-line distance is admissible). In other words, $h$ is an \textit{optimistic} estimate of the costs that actually occur.
A* Search from Arad to Bucharest

(a) The initial state

(b) After expanding Arad

(c) After expanding Sibiu

(d) After expanding Rimnicu Vilcea
A* Search from Arad to Bucharest

(e) After expanding Fagaras

(f) After expanding Pitesti
Example: Path Planning for Robots in a Grid-World
**Claim:** The first solution found has the minimum path cost.

**Proof:** Suppose there exists a goal node $G$ with optimal path cost $f^*$, but $A^*$ has found another node $G_2$ with $g(G_2) > f^*$. 

![Graph diagram](image-url)
Let \( n \) be a node on the path from the start to \( G \) that has not yet been expanded. Since \( h \) is admissible, we have

\[
f(n) \leq f^*.
\]

Since \( n \) was not expanded before \( G_2 \), the following must hold:

\[
f(G_2) \leq f(n)
\]
and

\[
f(G_2) \leq f^*.
\]

It follows from \( h(G_2) = 0 \) that

\[
g(G_2) \leq f^*.
\]

→ Contradicts the assumption!
Completeness:

If a solution exists, A* will find it provided that (1) every node has a finite number of successor nodes, and (2) there exists a positive constant $\delta > 0$ such that every step has at least cost $\delta$.

$\rightarrow$ there exists only a finite number of nodes $n$ with $f(n) \leq f^*$.

Complexity:

In general, still exponential in the path length of the solution (space, time).

More refined complexity results depend on the assumptions made, e.g. on the quality of the heuristic function. Example:

In the case in which $|h^*(n) - h(n)| \leq O(\log(h^*(n)))$, only one goal state exists, and the search graph is a tree, a sub-exponential number of nodes will be expanded [Gaschnig, 1977, Helmert & Roeger, 2008].
Heuristic Function Example

\[ h_1 = \text{the number of tiles in the wrong position} \]
\[ h_2 = \text{the sum of the distances of the tiles from their goal positions (Manhattan distance)} \]
\( d = \) distance from goal

- Average over 100 instances

<table>
<thead>
<tr>
<th>( d )</th>
<th>Search Cost (nodes generated)</th>
<th>Effective Branching Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IDS</td>
<td>( A^*(h_1) )</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>112</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>680</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>6384</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>47127</td>
<td>93</td>
</tr>
<tr>
<td>12</td>
<td>3644035</td>
<td>227</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>539</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>1301</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>3056</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>7276</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>18094</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>39135</td>
</tr>
</tbody>
</table>
Variants of $A^*$

$A^*$ in general still suffers from exponential memory growth. Therefore, several refinements have been suggested:

- iterative-deepening $A^*$, where the $f$-costs are used to define the cutoff (rather than the depth of the search tree): $IDA^*$
- Recursive Best First Search (RBFS): introduces a variable $f_{\_limit}$ to keep track of the best alternative path available from any ancestor of the current node. If current node exceeds this limit, recursion unwinds back to the alternative path.
- other alternatives memory-bounded $A^*$ ($MA^*$) and simplified $MA^*$ ($SMA^*$).
Local Search Methods

- In many problems, it is unimportant how the goal is reached—only the goal itself matters (8-queens problem, VLSI Layout, TSP).
- If in addition a quality measure for states is given, a **local search** can be used to find solutions.
- It operates using a single current node (rather than multiple paths)
- It requires little memory
- Idea: Begin with a randomly-chosen configuration and improve on it step by step → **Hill Climbing**.
- Note: It can be used for maximization or minimization respectively (see 8-queens example)
Example state with heuristic cost estimate $h = 17$ (counts the number of pairs threatening each other directly or indirectly).
function HILL-CLIMBING(problem) returns a state that is a local maximum

    current ← MAKE-NODE(problem.INITIAL-STATE)

loop do
    neighbor ← a highest-valued successor of current
    if neighbor.VALUE ≤ current.VALUE then return current.STATE
    current ← neighbor
Possible realization of a hill-climbing algorithm:
Select a column and move the queen to the square with the fewest conflicts.
Problems with Local Search Methods

- **Local maxima:** The algorithm finds a sub-optimal solution.
- **Plateaus:** Here, the algorithm can only explore at random.
- **Ridges:** Similar to plateaus but might even require suboptimal moves.

**Solutions:**

- *Start over* when no progress is being made.
- “Inject noise” $\rightarrow$ random walk

Which strategies (with which parameters) are successful (within a problem class) can usually only empirically be determined.
Local minimum \((h = 1)\) of the 8-queens Problem. Every successor has a higher cost.
Illustration of the ridge problem

The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima, that are not directly connected to each other. From each local maximum, all the available actions point downhill.
The 8-queens problem has about $8^8 \approx 17$ million states. Starting from a random initialization, hill-climbing directly finds a solution in about 14% of the cases. On average it requires only 4 steps!

Better algorithm: Allow sideways moves (no improvement), but restrict the number of moves (avoid infinite loops!).

E.g.: max. 100 moves: Solves 94%, number of steps raises to 21 steps for successful instances and 64 for failure cases.
Simulated Annealing

In the simulated annealing algorithm, “noise” is injected systematically: first a lot, then gradually less.

```plaintext
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
    inputs: problem, a problem
             schedule, a mapping from time to “temperature”

    current ← MAKE-NODE(problem.INITIAL-STATE)
    for t = 1 to ∞ do
        T ← schedule(t)
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← next.VALUE − current.VALUE
        if ΔE > 0 then current ← next
        else current ← next only with probability e^{ΔE/T}
```

Has been used since the early 80’s for VSLI layout and other optimization problems.
Evolution appears to be very successful at finding good solutions.

**Idea:** Similar to evolution, we search for solutions by three operators: “mutation”, “crossover”, and “selection”.

**Ingredients:**
- Coding of a solution into a string of symbols or bit-string
- A fitness function to judge the worth of configurations
- A population of configurations

**Example:** 8-queens problem as a chain of eight numbers. Fitness is judged by the number of non-attacks. The population consists of a set of arrangements of queens.
Selection, Mutation, and Crossing

Many variations:
how selection will be applied, what type of cross-over operators will be used, etc.

Selection of individuals according to a fitness function and pairing

Calculation of the breaking points and recombination

According to a given probability elements in the string are modified.
Summary

- **Heuristics** focus the search
- **Best-first search** expands the node with the highest worth (defined by any measure) first.
- With the minimization of the evaluated costs to the goal $h$ we obtain a greedy search.
- The minimization of $f(n) = g(n) + h(n)$ combines uniform and greedy searches. When $h(n)$ is admissible, i.e., $h^*$ is never overestimated, we obtain the $A^*$ search, which is complete and optimal.
- **IDA** is a combination of the iterative-deepening and $A^*$ searches.
- **Local search methods** only ever work on one state, attempting to improve it step-wise.
- **Genetic algorithms** imitate evolution by combining good solutions.