Informed Search Strategies

Here is the reformatted document with improved readability:

Week 4-5: Informed Search Strategies

Dr. Ammar Masood

Department of Cyber Security, Air University Islamabad

Spring 2025 CS340 - Introduction to AI

© Dept of Cyber Security

Contents

- Informed Strategies
- Greedy Best-First Search
- A* Search
- Weighted A*
- Recursive Best-First Search

Quick Review

Previous algorithms differed in how they selected the next node for expansion:

- **Breadth-First Search** → Frontier nodes sorted from oldest to newest
- **Depth-First Search** → Frontier nodes sorted from newest to oldest
- **Uniform-Cost Search** → Frontier nodes sorted by path cost (smallest to largest)

These algorithms used little to no external domain knowledge.

Informed Strategies

Informed search strategies use domain-specific hints about goal locations. These hints come in the form of **heuristic functions** $h(n)$ that help guide the search process more efficiently.

$$h(n) = \text{estimated cost of the cheapest path from state at node } n \text{ to a goal state}$$

Greedy Best-First Search

- Expands the node that appears closest to the goal, assuming it leads to a solution quickly.
- **Evaluation function**:

$$f(n) = h(n)$$

- **Implementation:**
  - Expands the most desirable node first.
  - The frontier queue is sorted in decreasing order of desirability.
- **Example:** Using the straight-line distance heuristic h_{SLD}, the algorithm expands nodes appearing closest to the goal.
- **Evaluation function:**

$$f(n) = h_{SLD}(n)$$

Exercise

Is the path Arad → Sibiu → Fagaras → Bucharest optimal?

The path via Sibiu and Fagaras to Bucharest is **32 miles (450 miles total)** longer than the path through Rimnicu Vilcea and Pitesti (418 miles). This is why the algorithm is called "greedy"—it chooses what seems best at the moment, even if it is not optimal.

Greedy Best-First Search Analysis

- **Not optimal**
- **Not complete** (May go down an infinite loop without reaching the goal)
  - Example: $\text{Iasi} \to \text{Neamt} \to \text{Iasi} \to \text{Neamt} \to \dots$
- **Space Complexity:** $O(b^m)$ (Keeps all nodes in memory)
- **Time Complexity:** $O(b^m)$, but a good heuristic can improve performance.

*A Search**

A* combines **Uniform-Cost Search** and **Greedy Best-First Search** using the evaluation function:

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

where:

- $g(n)$ = cost from the start node to node n
- $h(n)$ = estimated cost from n to the goal
- $f(n)$ = estimated total cost from start to goal through n

*A Search Properties**

- If $h(n)$ is the actual cost to the goal → **Expands only nodes on the correct path** (Optimal solution)
- If $h(n) \leq$ actual cost to goal → **Admissible heuristic** (Expands additional nodes but still finds the optimal solution)
- If $h(n) >$ actual cost to goal → **Optimal solution may be overlooked**

Admissibility and Consistency

- *A is optimal if it uses an admissible heuristic**, meaning:

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

  where $h^*(n)$ is the true cost from node n to the goal.

- **Consistency (Monotonicity):**

  $$h(n) \leq c(n, a, n') + h(n')$$

  - Ensures that the estimated cost never increases along a path.
  - Every **consistent heuristic** is **admissible** (but not vice versa).

Contours in A*

A* expands nodes in increasing order of f value, forming contours like topographic maps:

- **Inside contour** labeled **400** → All nodes have $f(n) = g(n) + h(n) \leq 400$
- *Contours show how A focuses on the goal rather than spreading equally**

Key Points

- *A expands all nodes where $f(n) < C*$** (optimal solution cost).
- **Some nodes on the "goal contour" $f(n) = C*$ may be expanded before reaching the goal.**

- **No nodes with $f(n) > C*$ are expanded.**

*A Search Complexity**

- **Complete?** Yes, unless there are infinite nodes with $f \leq f(G)$.
- **Time Complexity?**
    - Best case: $O(d)$ if $h(n)$ is perfect
    - Worst case: $O(b^d)$ (same as BFS)
- **Space Complexity?** $O(b^d)$ (Stores all nodes in memory)

*A usually runs out of space before it runs out of time.**

Satisficing Search

- Aims to find a **"good enough"** solution rather than the optimal one.
- Useful when **optimality is computationally expensive** or unnecessary.

Weighted A*

A variation of A* that **prioritizes heuristic guidance** more heavily:

$$f(n) = g(n) + w \cdot h(n)$$

where:

- $g(n)$ = Cost from the start node to n
- $h(n)$ = Heuristic estimate to the goal
- w = Weight factor ($w > 1$), increasing heuristic influence

Comparison of Search Strategies

Search Strategy	Cost Function	Weight (W)
A*	$g(n) + h(n)$	$W = 1$
Uniform-Cost Search	$g(n)$	$W = 0$
Greedy Best-First Search	$h(n)$	$W = \infty$
Weighted A*	$g(n) + W \times h(n)$	$1 < W < \infty$

Memory-Bounded Heuristic Search

- **Iterative Deepening A*** (IDA*): Like IDS but cuts off when $f(n) >$ max.
- **Recursive Best-First Search (RBFS):** Uses recursion with limited memory.
- **Simple Memory Bounded A*** (SMA*): Uses a memory limit, dropping the worst node if memory is full.

Recursive Best-First Search (RBFS)

- Works like depth-first search but **tracks the best alternative path**.
- Uses **backtracking** when the current node exceeds the f-limit.
- Requires **less memory than A*** because it **does not store all explored nodes**.

Applications of Search Algorithms

Search Algorithm	**Type**	**Real-World Application**
Uniform-Cost Search	Uninformed	Finding the least costly path in transportation networks
Bidirectional Search	Uninformed	Network routing optimization
Greedy Best-First Search	Informed	Web crawling, information retrieval
A*	Informed	Robotics navigation, AI pathfinding
Weighted A*	Informed	Game development, balancing speed vs safety
Recursive Best-First Search	Informed	Solving complex puzzles, chess strategies

---

This structured formatting improves readability while preserving all original content. Let me know if you need further refinements! 🚀