INFORMED (HEURISTIC) SEARCH STRATEGIES

Artificial Intelligence:Unit II

Lecture Outline

Introduction.	Best First.	Greedy.
A*.	AO*.	Hill Climbing.

Introduction

How an informed strategy—one that uses problem-specific knowledge beyond the definition of the problem itself—can find solutions more efficiently than can an uninformed strategy.

Best-First Search

- An instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function, f(n).
- The evaluation function is construed as a cost estimate, so the node with the lowest evaluation is expanded first.
- The implementation is identical to that for uniform-cost search, except for the use of **f** instead of **g** to order the priority queue.
- The choice of f determines the search strategy.
- Most f a heuristic function, denoted h(n):
- h(n) = estimated cost of the cheapest path from the state at node n to a goal state.
- (Notice that h(n) takes a node as input, but, unlike g(n), it depends only on the state at that node.)
- For now, consider h(n) to be arbitrary, nonnegative, problem-specific functions, with one constraint: if n is a goal node, then h(n)=0.

Greedy Best-First Search

- Tries to expand the node that is closest to the goal, which lead to a solution quickly.
- Evaluates nodes by using the heuristic function; f(n) = h(n).
- **Route-finding problems in Romania;** use the straight line distance heuristic, (h_{SLD}).
- **Goal** is Bucharest, need to know the straight-line See table.
- h_{SLD}(In(Arad))=366.
- h_{SLD} not be computed from the problem description. It takes a certain amount of experience to know that h_{SLD} is correlated with actual road distances.
- A path from Arad to Bucharest.
- first node to be expanded will be **Sibiu**, closer to Bucharest than either Zerind or Timisoara.
- Next node to be expanded will be **Fagaras** because closest. Fagaras in turn generates Bucharest, which is the goal.
- Here finds a solution without ever expanding a node that is not on the solution path; **=> search cost is minimal**.
- It is not optimal, however: the path via Sibiu and Fagaras to Bucharest is 32 kilometers longer than the path through Rimnicu Vilcea and Pitesti.
- Shows why the algorithm is called "greedy"—at each step it tries to get as close to the goal as it can.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Values of h_{SLD}—straight-line distances to Bucharest.

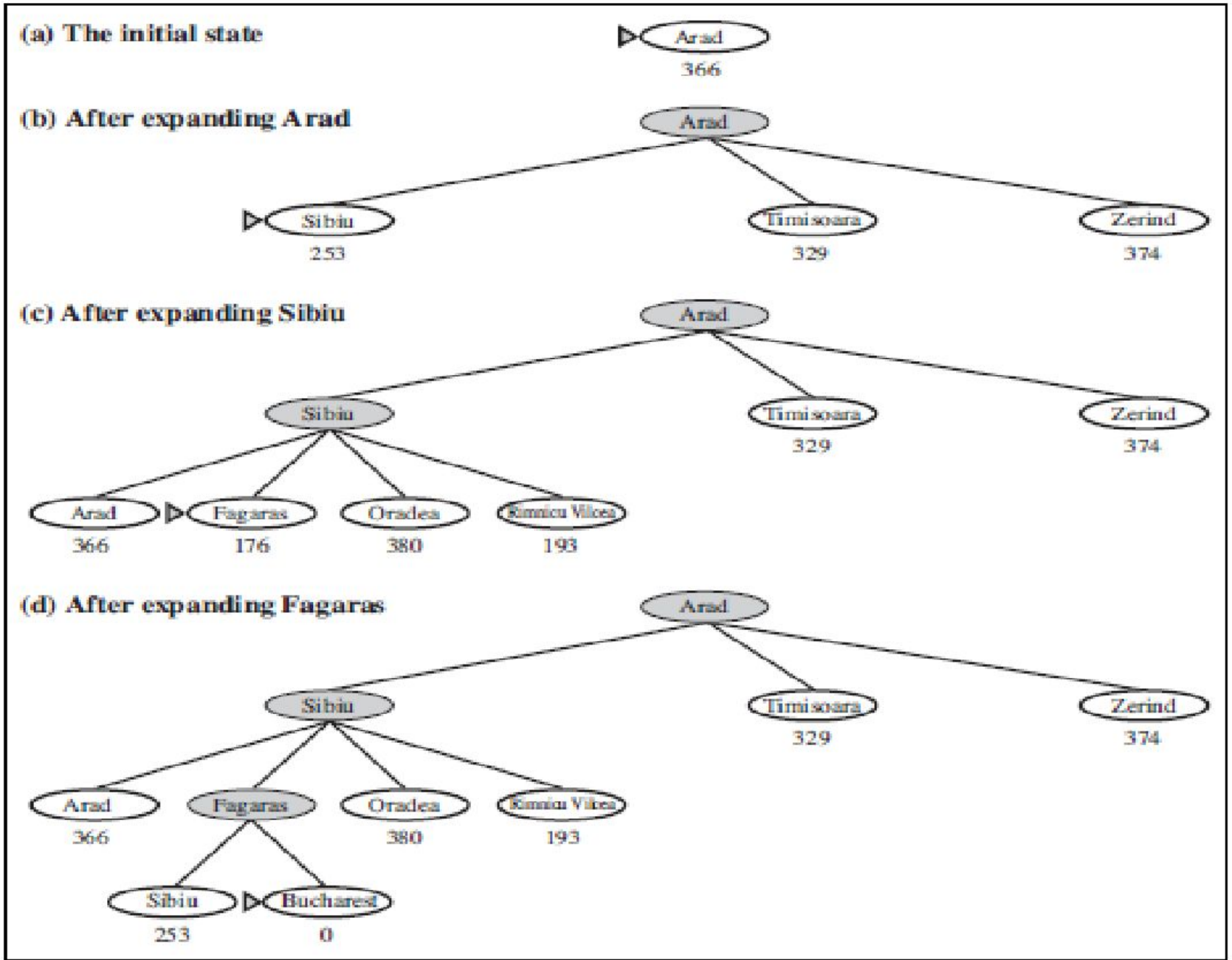
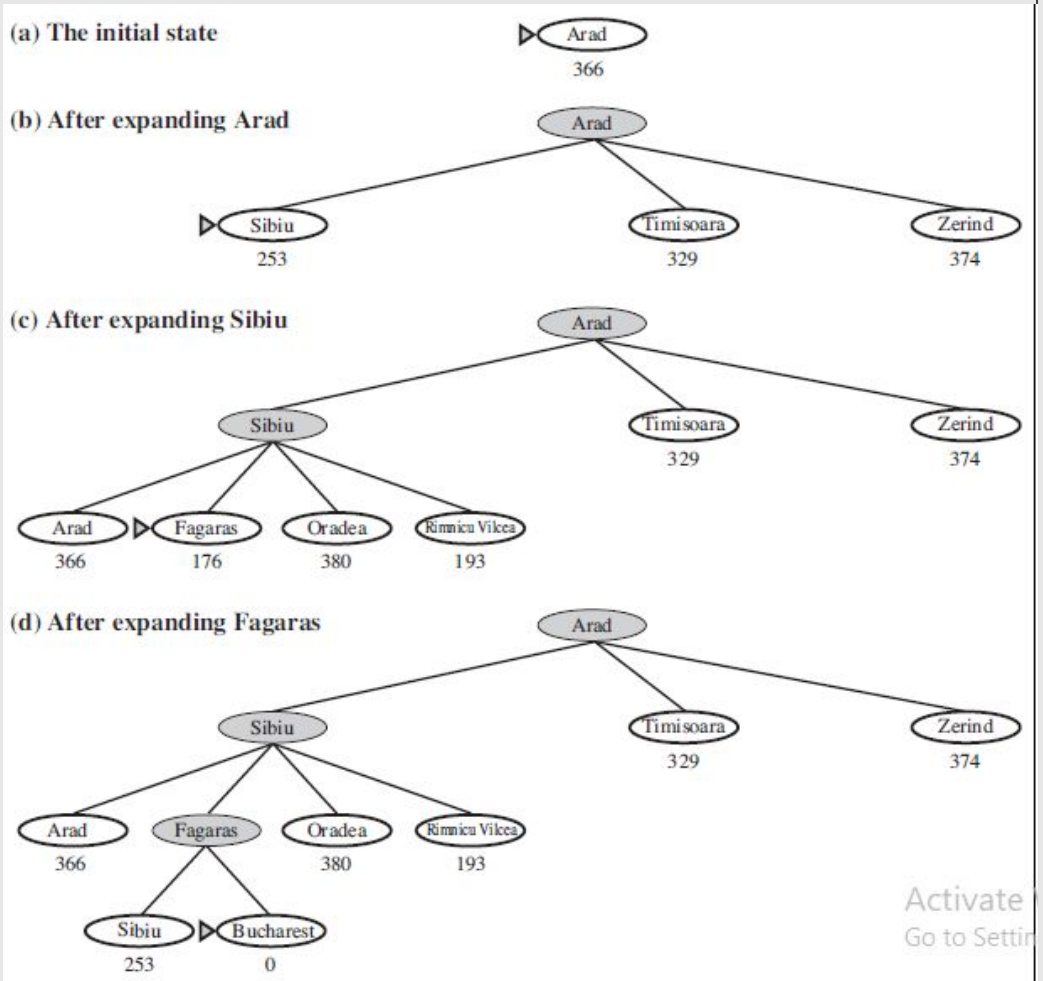


Figure 3.23 Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic h_{SLD}. Nodes are labeled with their h-values.

- The most widely known form of best-first search is called A* A search
- It evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal: $f(n) = g(n) + h(n)$.
- Here, g(n) = path cost from the start node to node n, h(n)= estimated cost of the cheapest path from n to the goal, therefore f(n) = estimated cost of the cheapest solution through n .
- Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of g(n) + h(n).
- It turns out that this strategy is more than just reasonable: provided that the heuristic function h(n) satisfies certain conditions, A* search is both complete and optimal.
- The algorithm is identical to UNIFORM-COST-SEARCH except that A* uses g + h instead of g.

A* search: Minimizing the total estimated solution cost



- **Conditions for optimality: Admissibility and consistency**
- The first condition we require for optimality is that $h(n)$ be an **admissible heuristic**.
- An admissible heuristic is one that *never overestimates* the cost to reach the goal. Because $g(n)$ is the actual cost to reach n along the current path, and $f(n)=g(n) + h(n)$, we have as an immediate consequence that $f(n)$ never overestimates the true cost of a solution along the current path through n.
- Admissible heuristics are by nature optimistic because they think the cost of solving the problem is less than it actually is.
- An obvious example of an admissible heuristic is the straight-line distance h_{SLD} that we used in getting to Bucharest. Straight-line distance is admissible because the shortest path between any two points is a straight line, so the straight line cannot be an overestimate.
- A second, slightly stronger condition called **consistency** (or sometimes **monotonicity**)
- Required only for applications of A* to graph search. A heuristic $h(n)$ is consistent if, for every node n and every successor n' of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n':

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

- This is a form of the general **triangle inequality**, which stipulates that each side of a triangle

cannot be longer than the sum of the other two sides.

Here, the triangle is formed by n, n' and the goal G_n closest to n.

For an admissible heuristic, the inequality makes perfect sense:

if there were a route from n to G_n via n' that was cheaper than h(n), that would violate the property that h(n) is a lower bound on the cost to reach G_n.

- every consistent heuristic is also admissible.
- Consistency is therefore a stricter requirement than admissibility, but one has to work quite hard to concoct heuristics that are admissible but not consistent.
- Consider, for example, h_{SLD}. We know that the general triangle inequality is satisfied when each side is measured by the straight-line distance and that the straight-line distance between n and n' is no greater than c(n, a, n').

Hence, h_{SLD} is a consistent heuristic.

- **Conditions for optimality: Admissibility and consistency**
- The first condition we require for optimality is that $h(n)$ be an **admissible heuristic**.
- An admissible heuristic is one that *never overestimates* the cost to reach the goal. Because $g(n)$ is the actual cost to reach n along the current path, and $f(n)=g(n) + h(n)$, we have as an immediate consequence that $f(n)$ never overestimates the true cost of a solution along the current path through n.
- Admissible heuristics are by nature optimistic because they think the cost of solving the problem is less than it actually is.
- An obvious example of an admissible heuristic is the straight-line distance h_{SLD} that we used in getting to Bucharest. Straight-line distance is admissible because the shortest path between any two points is a straight line, so the straight line cannot be an overestimate.
- A second, slightly stronger condition called **consistency** (or sometimes **monotonicity**)
- Required only for applications of A* to graph search. A heuristic $h(n)$ is consistent if, for every node n and every successor n' of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n':

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

- This is a form of the general **triangle inequality**, which stipulates that each side of a triangle

cannot be longer than the sum of the other two sides.

Here, the triangle is formed by n, n' and the goal G_n closest to n.

For an admissible heuristic, the inequality makes perfect sense:

if there were a route from n to G_n via n' that was cheaper than h(n), that would violate the property that h(n) is a lower bound on the cost to reach G_n.

- every consistent heuristic is also admissible.
- Consistency is therefore a stricter requirement than admissibility, but one has to work quite hard to concoct heuristics that are admissible but not consistent.
- Consider, for example, h_{SLD}. We know that the general triangle inequality is satisfied when each side is measured by the straight-line distance and that the straight-line distance between n and n' is no greater than c(n, a, n').

Hence, h_{SLD} is a consistent heuristic.

- **Optimality of A***
- A* has the following properties: *the tree-search version of* A* *is optimal if* h(n) *is admissible, while the graph-search version is optimal if* h(n) *is consistent.*
- We show the second of these two claims since it is more useful. The argument essentially mirrors the argument for the optimality of uniform-cost search, with g replaced by f—just as in the A* algorithm itself.
- The first step is to establish the following: *if* h(n) *is consistent, then the values of*f(n) *along any path are nondecreasing.* The proof follows directly from the definition of consistency.
- Suppose n' is a successor of n; then g(n')=g(n) + c(n, a, n') for some action a, and we have f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') ≥ g(n) + h(n) = f(n) .
- The next step is to prove that *whenever* A* *selects a node* n *for expansion, the optimal path to that node has been found.* Were this not the case, there would have to be another frontier node n' on the optimal path from the start node to n, by the graph separation property of …….**refer book now.**