Solving Problems by Searching

Informed (Heuristic) Search Strategies

This section shows how an **informed (heuristic) search** strategy one that uses problem-specific knowledge beyond the definition of the problem itself can to find solutions more efficiently than can an uninformed strategy. Heuristic functions are the most common form in which additional knowledge of the problem is imparted to the search algorithm. The remainder of this section covers the ways to use heuristic information to guide search algorithm.

1. **Greedy best-first search:** greedy best-first search tries to expand the node that is closest to the goal, on the grounds that is likely to lead to a solution quickly. Thus, it evaluates nodes by using just the heuristic function, f(n) = h(n).

   Let us see how this works for the route-finding problems in Romania; we use the Straight-line distance heuristic, which we will call h_{SLD}. If the goal is Bucharest, we need to know the straight-line distance to Bucharest, which are shown in figure below.

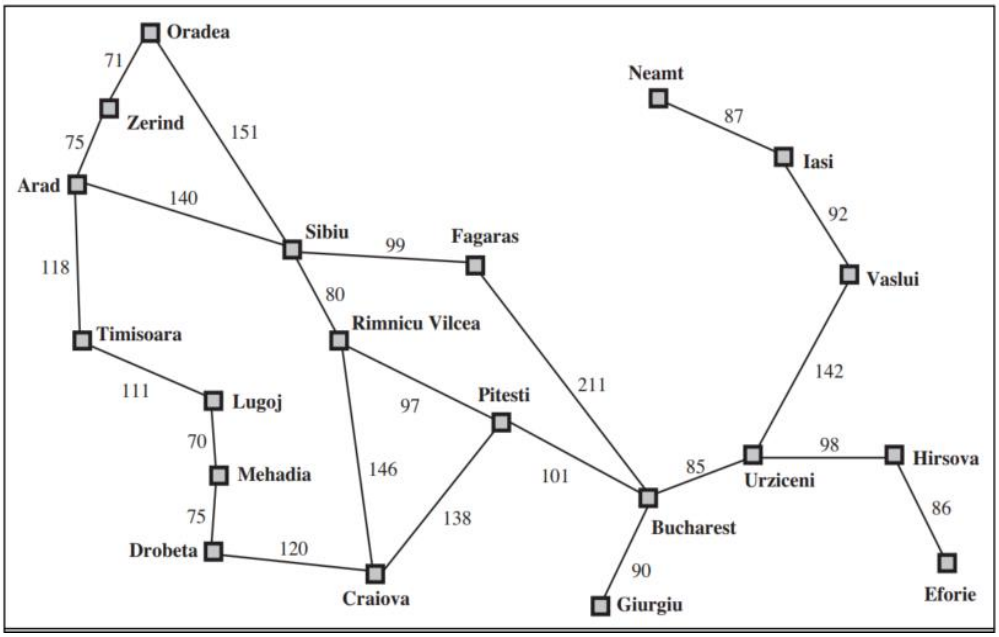
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Figure 3.22 Values of h_{SLD}—straight-line distances to Bucharest.

   For example, h_{SLD}(In(Arad)) = 366. Notice that the values of h_{SLD} cannot be computed from the problem description itself. Moreover, it takes a certain amount of experience to know that h_{SLD} is correlated with actual road distance and is, therefore, a useful heuristic.

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

Figure below shows the progress of greedy best first-search using h_{SLD} to find a path from Arad to Bucharest.



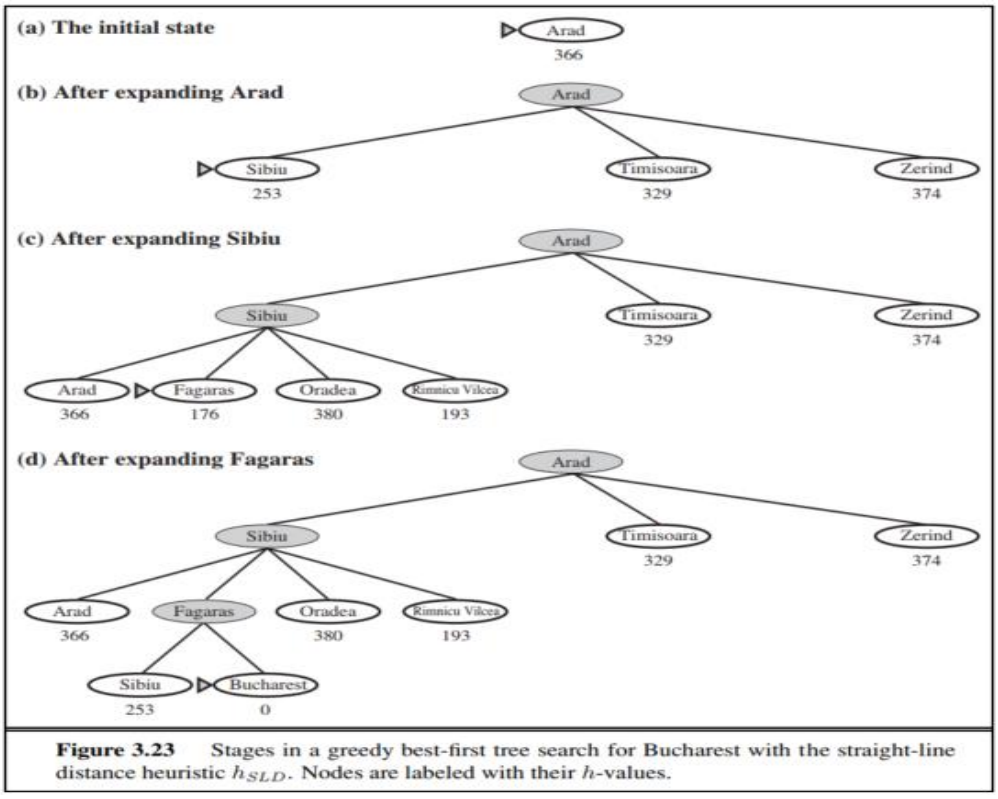


Figure 3.23 Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic h_{SLD}. Nodes are labeled with their h-values.

The first node to expanded from Arad will be Sibiu because is closer to Bucharest than either Zerind or Timisoara. The next node to expanded will be Fagaras because it is closest. Fagaras in turn generates Bucharest, which is the goal. For this particular problem, greedy best-first search using h_{SLD} finds a solution without ever expanding a node that is not on the solution path. Hence, its search cost is minimal. It is not optimal, however: the path via Sibiu and Fagaras to Bucharest is 32 kilometers longer that the path through Rimnicu Vilcea and Pitesti. This show why the algorithm is called greedy at each step it tries to get as close to the goal as it can.

Greedy best-first tree search is also incomplete even in a finite state space, much like depth-first search. Consider the problem of getting from Iasi to Fagaras. The heuristic suggests that Neamt be expanded first because it is closest fo Fagaras, but it is a dead end. The solution is to go first to Vaslui a step that is actually farther from the goal according the heuristic and then to continue to Urziceni, Bucharest, and Fagaras. The algorithm will never find this solution, however, because expanding Neamt puts Iasi back inot the frontier, Iasi is closer to Fagaras than Vaslui is, and so Iasi will be expanded again, leading to an infinite loop.

2. **A* Search:** A* search 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:

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

since g(n) gives the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have

f(n) = estimate 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 jus 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.

In figure below, we show the progress of an A* search for Bucharest. The Values of g and h_{SDL} are taken from the previous example.

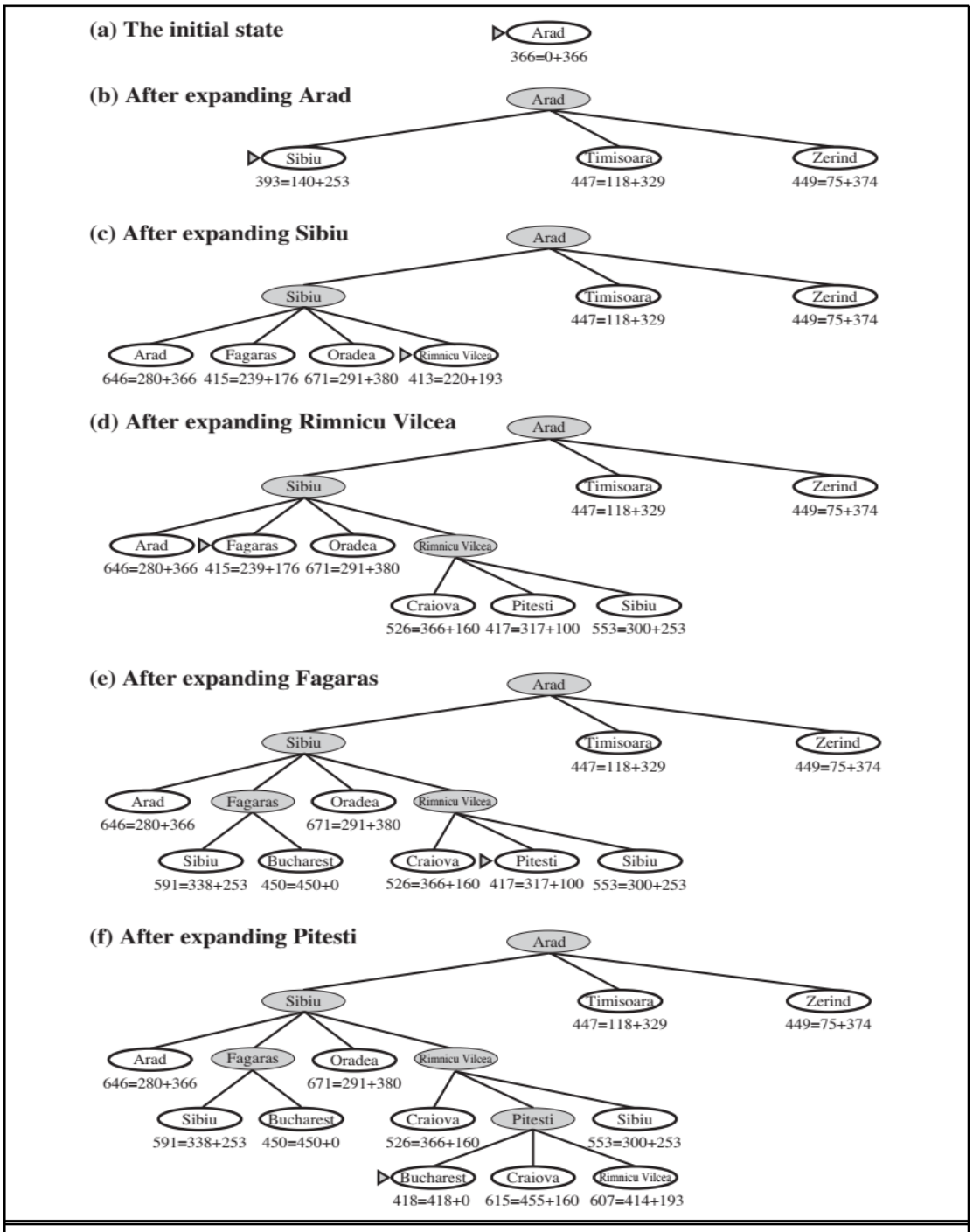


Figure 3.24 Stages in an A* search for Bucharest. Nodes are labeled with $f = g + h$. The h values are the straight-line distances to Bucharest taken from Figure 3.22.

Notice in particular that Bucharest first appears on the frontier at step (e), but it is not selected for expansion because its f -cost(450) is higher than that of Pitesti (417). Another way to say this is that there might be a solution through Pitesti whose cost is as low as 417, so the algorithm will not settle for a solution that cost (450).