4

ARTIFICIAL INTELLIGENCE
COM S 472/572 LECTURE NOTES
WEEK 4

Author: Vasant Honavar
Scribes: D. Keller, S. Laird, A. Lamba

Computer Science Department,
Iowa State University, Ames, IA 50011

1 HEURISTIC SEARCH (CONTINUED)

This brings us to our goal of arriving at an optimal heuristic search algorithm; one that finds an optimal (cheapest or of highest quality) solution with minimal search effort given the information provided. To enable us to do so, we first have to quantify things like the information available, the quality of a solution, and the search effort.

By combining a number of heuristics, it is possible to create a new heuristic that comes closer to the actual cost function $h^*(n)$. A* is such a search.

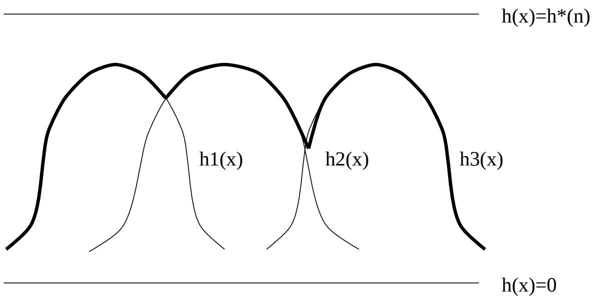


Figure 1 Multiple Heuristics

1.1 A* Algorithm

By combining the *Best-First Heuristic Search*, using the estimated cost of completing a partial solution as measured by h values, and the *Branch-and-Bound Search*, where we make use of the cost information, we get the A* Algorithm. This can be seen in the evaluation function of the A* Algorithm $f(n) = g(n) + h(n)$ where $g(n)$ represents the known least-cost path from s to n, $h(n)$ is the estimated least-cost path from n to a goal node, and $f(n)$ represents the estimated cost of the path from the start node s to a goal node g that is constrained to pass through n.

Essentially, A* is aimed to provide an algorithm for finding an optimal solution with minimal effort. It is basically a refinement of B&B (Branch-and-Bound) search with dynamic programming by adding guesses (or heuristic estimates) of the cost of reaching a goal node from the current state by extending the current partial path. The estimated total path length for such a search would be the distance already traveled plus the estimate of remaining distance (measured by "h").

The basic idea of A* is to always extend the partial path with the cheapest estimated path length.

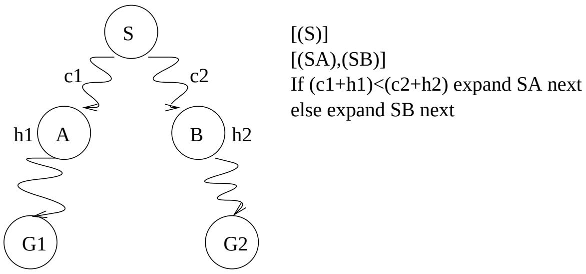


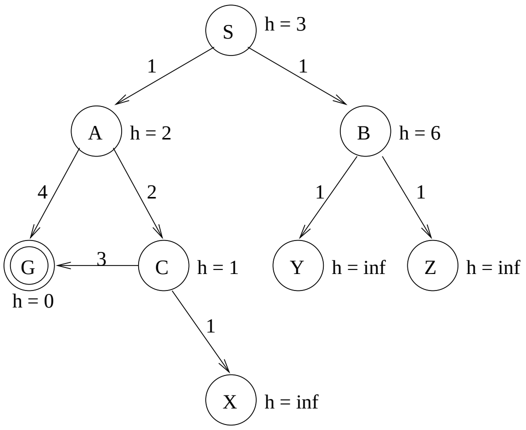
Figure 2 Graph of Cheapest Partial Path

1.2 Pseudo Code for A*

```
1.  Initialize $L$ to a path of length $0$ containing $s$
2.  If the list $L$ is empty return FAILURE
3.  Pick the node $x \epsilon L$ that has the least $f$ value
    among all nodes in $L$
4.  If $x \epsilon G$ (where $G$ is the goal set)
      then return the path $s \ldots x$ and stop
      else expand $x$ and place its children on $L$
5.  Go to step 2
```

Figure 3 Pseudo Code for A*

1.3 Example of A*



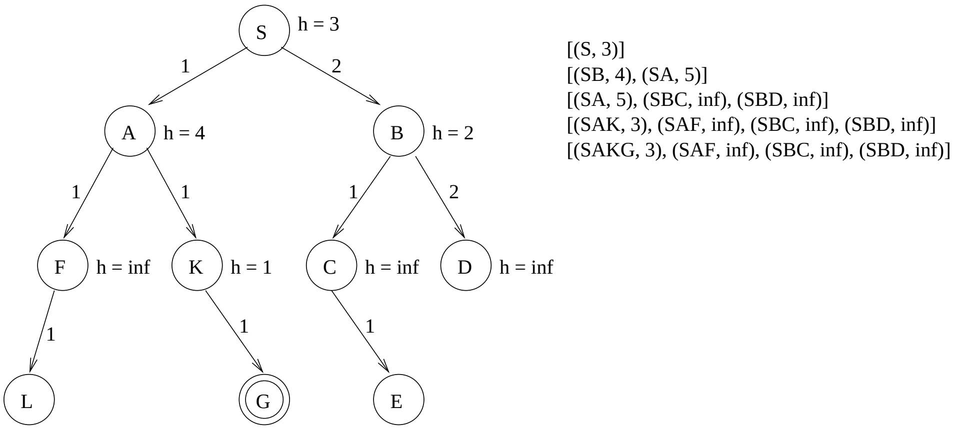
(path, f(n))
[(S, 3)]
[(SA, 3), (SB, 7)]
[(SAC, 4), (SAG, 5), (SB, 7)]
[(SAG, 5), (SACG,6), (SB, 7), (SACX, inf)]

Figure 4 Graph of Example using A*

1.4 Exercises using A*

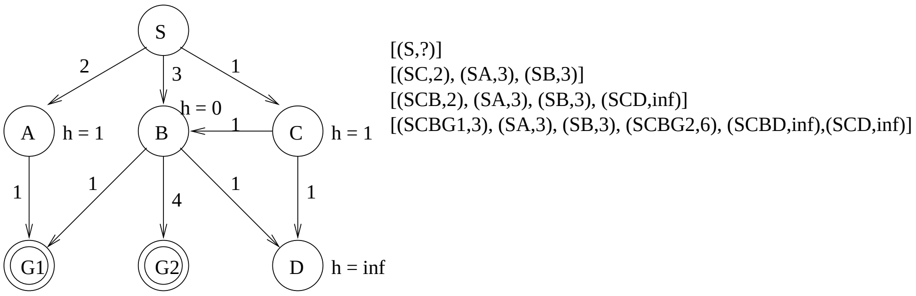
Exercise 1: Write down the list of partial path maintained by A*:

Exercise 2: Calculate the cost of finding the goal node. f(n) = g(n) + h(n)
where g(n) = cost of the cheapest known path to n;
and h(n) = estimated cheapest cost of going from n to a goal.



[(S, 3)]
[(SB, 4), (SA, 5)]
[(SA, 5), (SBC, inf), (SBD, inf)]
[(SAK, 3), (SAF, inf), (SBC, inf), (SBD, inf)]
[(SAKG, 3), (SAF, inf), (SBC, inf), (SBD, inf)]

Figure 5 Graph for Exercise 1



[(S,?)]
[(SC,2), (SA,3), (SB,3)]
[(SCB,2), (SA,3), (SB,3), (SCD,inf)]
[(SCBG1,3), (SA,3), (SB,3), (SCBG2,6), (SCBD,inf),(SCD,inf)]

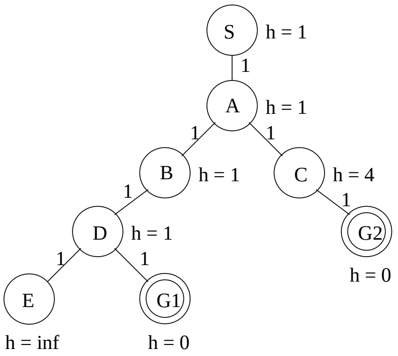
Figure 6 Graph for Exercise 2

1.5 Questions about A* Algorithm

1. Q. What can we say about A* if *h* is a perfect heuristic?
   A. There is essentially no search. The algorithm focuses on the best path.

2. Q. What if *h* overestimates the actual cost (of reaching a goal)?
   A. There is a likelihood that one may miss an optimal path (in favor of a worse path).

3. Q. What if *h* is guaranteed to be an underestimate (of the actual costs of reaching a goal)?
   A. We will prove that in this case, A* is guaranteed to find an optimal path (if a path exists).



[(S, 1)]
[(SA, 2)]
[(SAB, 3), (SAC, 6)]
[(SABD, 4), (SAC, 6)]
[(SABDG1, 4), (SAC, 6), (SABDE, inf)]

As a result of over-estimation of h,
G1 will be found instead of G2.

Figure 7 Graph with Overestimate

1.6 Ground Work for Proofs

We will attempt to prove that if the h function is an underestimate, an optimal path would have been found when the search is terminated.

Examples of Underestimate

1. In trying to find the shortest path between two cities by road, a heuristic function which is an underestimate is the straight line distance between the two cities.
2. In the puzzle problem with 9 positions and 8 numbered tiles, the number of tiles in the incorrect position is a heuristic function which is an underestimate.

The evaluation function of A* Algorithm is f(n) = g(n) + h(n) where g(n) represents the cost of the cheapest known path from s to n and h(n) is the estimated least-cost path from n to a goal node (h(n) has to be an underestimate relative to the cost of the cheapest path).

Without loss of generality, we assume h(n) and g(n) to be greater or equal to 0 $\forall$n in the state space.

Definition. A search algorithm is said to be complete if it is guaranteed to terminate with a solution whenever a solution exists.

To arrive at a search that produces optimal solutions, *completeness* is inadequate. Thus we reach the following stronger definition of what it means for a search to be *admissible*.

Definition. A search algorithm is said to be admissible if it is guaranteed to terminate with an optimal (i.e., cheapest) solution whenever a solution exists.

Search	Complete	Admissible
Depth First	No	No
Breadth First	Yes	No (in general) Yes (for uniform cost arcs)
IDS	Yes	No (in general) Yes (for uniform cost arcs)
BBS	Yes	Yes
Best First	No	No
A^*	Yes (for locally finite graphs)	Yes (provided $h(n)$ is an underestimate)

With that in mind, we set out to prove that A^* is admissible.

1.7 Admissibility of A^*

Theorem. A^* is admissible (i.e. it is guaranteed to terminate with an optimal solution if a solution exists) if:

1. the branching factor is finite.
2. for all n_i, n_j in Ψ, $k(n_i, n_j) \geq \delta > 0$ (i.e. all costs are positive). So $k(n_i, n_j)$ is the actual cost of the minimum cost path from n_i to n_j (infinite or undefined if no path).
3. for all n in Ψ, $h(n) \leq h^*(n)$ (i.e. all heuristics are underestimates).

Evaluation function — f(n) = g(n) + h(n) where
g(n) = cost of cheapest known path from s to n;
h(n) = estimate of cheapest cost from n to a goal;
f(n) = estimated cost of cheapest path from s to g through n.

$c(n_i, n_j)$ - cost of an arc connecting node n_i to n_j $\boxed{c(n_i, n_j) \geq \delta > 0}$

The actual cost of the minimum cost path from n to a goal g_i = k(n, g_i) and $h^*(n) = \underset{g_i \in G}{min} \{k\ (n,\ g_i)\ \}$

Clearly, any path from n to a goal in G with cost = $h^*(n)$ is an optimal path from n to that goal.

Definition: Let $g^*(n) = k(s, n)\ \forall n$ accessible from s. Define $f^*(n) = g^*(n) + h^*(n)$ (The cost of the optimal path from s to a goal that is constrained to pass through node n).

f(n), g(n) and h(n) are estimates of $f^*(n)$, $g^*(n)$ and $h^*(n)$ respectively, where $f^*(n)$, $g^*(n)$ and $h^*(n)$ are optimal.

g(n) = sum of arc costs along the shortest known path from s to n and can be modified as we discover cheaper/better paths to n.

Now $g^*(n) \leq g(n)$.

In order to prove that A^* is *admissible*, we've to first prove that it is *complete*