IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 4, Issue 2 (Sep.-Oct. 2012), PP 01-05
www.iosrjournals.org

Heuristic Searching: A* Search

Nazmul Hasan
(Department of Computer Science and Engineering, Military Institute of Science and Technology, Dhaka-1216, Bangladesh)

Abstract: *Searching has a great impact on computer science. There are lots of searching algorithms. Among them, A* search algorithm is one of the most promising algorithm. A* search algorithm is more efficient than the other searching algorithm, because of its heuristic characteristic. A* search is widely used in path finding and graph traversal among the points, called nodes. Due to the heuristic characteristic A* search must find the goal with the minimum cost, if obviously there is a goal state. For this reason there are many application of this. The aim of this paper is to understand the A* search technique, its characteristics, mathematical representation, some graphical representation.*

Keywords*: Admissibility, Best-First Search, Consistency, Dominance, Heuristic.*

I. Introduction

Similar to the other informed search algorithms, A* search algorithm find the path that is most promising to reach the goal node. A* search algorithm use the Best-First Search technique to find the lowest cost path to find the goal node starting from the starting node. To traverse a graph A* search follows the path of the lowest known heuristic cost. And it also keeps a sorted priority queue of alternate path along the way. On the way to the solution, A* search maintain two list to keep count of the nodes that has been traversed and nodes to be traversed. There are referred as the open list and close list [1]. Open list keep track of the nodes that has been already traversed, and closed list keep track the nodes that will be traversed in future.

It uses a function that contains the summation the cost of distance and cost to demonstrate the order in which visits nodes in the tree. For this, it has to maintain an equation. Here the equation, Total cost = Cost of traversed from the initial node to current node + Cost of the path to be traverse for the goal node.

From the initial state A* search create a priority queue to store the nodes that to be explored. Every node is prioritized according to the summation of the cost of already traversed and the cost of the path that will traverse. The lower the value of summation, the higher the priority. Every step in this algorithm the node with the higher priority (lowest summation value) will be removed from the queue. In every step of movement the cost of the path that have been traversed and that is yet to traverse will update simultaneously. At the goal node the value of the summation of the cost of the already traversed and node to be traversed will be lower than any other node. I f there are more than one ways to go to the goal state, and then the path by which lowest summation value has obtained, will be selected. At the final state the cost of the node that is to be traverse will be zero (0).

II. Mathematical representation of A* search

Here the mathematical representation of the A* search technique.
The equation is

$$f(x) = g(x) + h(x).$$

Here,

$f(x)$ = The total distance (cost).
$g(x)$ = The distance (cost) from the initial position to the current position.
$h(x)$ = The heuristic function that is used to approximate distance (cost) from the current location to the goal state.

This function is distinct because it is a mere estimation rather than an exact value. The more accurate the heuristic the better the faster the goal state is reach and with much more accuracy. When starting at the initial node this search keeps track of the nodes to be traversed. This is known as the open set and it keeps track of nodes in order of the lowest ***f(x)*** to the highest ***f(x)***. In order of priority each node is removed from the queue and then the values for ***f(x)*** and ***h(x)*** neighboring nodes around the removed node are updated. This search continues until it reaches the node with the lowest ***f(x)*** value, which is called the goal node. **h(x)** at the goal is zero which is an admissible heuristic because the path to the goal is clearly not an overestimate.

If the heuristic function h(x) never overestimates the minimum cost of reaching the goal, also known as being admissible, then A* is also known to be admissible.

II.I Pseudo code

```
function A*(start, goal)
    closedset := the empty set                    // The set of nodes already evaluated.
    openset := {start}                            // The set of tentative nodes to be evaluated, initially containing the
start node
    came_from := the empty map                    // The map of navigated nodes.

    g_score[start] := 0                           // Cost from start along best known path.
                                                  // Estimated total cost from start to goal through y.
    f_score[start] := g_score[start] + heuristic_cost_estimate(start, goal)

    while openset is not empty
        current := the node in openset having the lowest f_score[] value
        if current = goal
            return reconstruct_path(came_from, goal)

        remove current from openset
        add current to closedset
        for each neighbor in neighbor_nodes(current)
            if neighbor in closedset
                continue
            tentative_g_score := g_score[current] + dist_between(current,neighbor)

            if neighbor not in openset or tentative_g_score < g_score[neighbor]
                add neighbor to openset
                came_from[neighbor] := current
                g_score[neighbor] := tentative_g_score
                f_score[neighbor] := g_score[neighbor] + heuristic_cost_estimate(neighbor, goal)

    return failure

function reconstruct_path(came_from, current_node)
    if came_from[current_node] is set
        p := reconstruct_path(came_from, came_from[current_node])
        return (p + current_node)
    else
        return current_node
```

[Pseudo code source: http://en.wikipedia.org/wiki/A*_search_algorithm [8]]

III. Table and Graphical Representation

Now for better understand, we will illustrate table along with graphical representation. Starts from node A.

III.I Table

Now at first we consider the path: A → F → H → I → D → E → G

Node	Value of g(x)	Value of h(x)	Value of f(x)
F	15	85	100
H	15	75	90
I	15	45	60
D	10	30	40
E	15	10	25
G	5	0	5

Table 1

Now we consider the second path: A → B → C → D → E → G

Node	Value of g(x)	Value of h(x)	Value of f(x)
B	10	75	85
C	10	70	80
D	10	30	40
E	15	10	25
G	5	0	5

Table 2

Now we consider the Third path: A → B → C → J → K → G

Node	Value of g(x)	Value of h(x)	Value of f(x)
B	10	75	85
C	10	70	80
J	12	55	67
K	15	27	42
G	5	0	5

Table 3

III.II Result

Now among the three possible ways to reach goal state from the start state, the second path that means,

A → B → C → D → E → G

will cost lower than the others. So, A* search will follow this path to reach the goal node G.

III.IV Graphical Representation

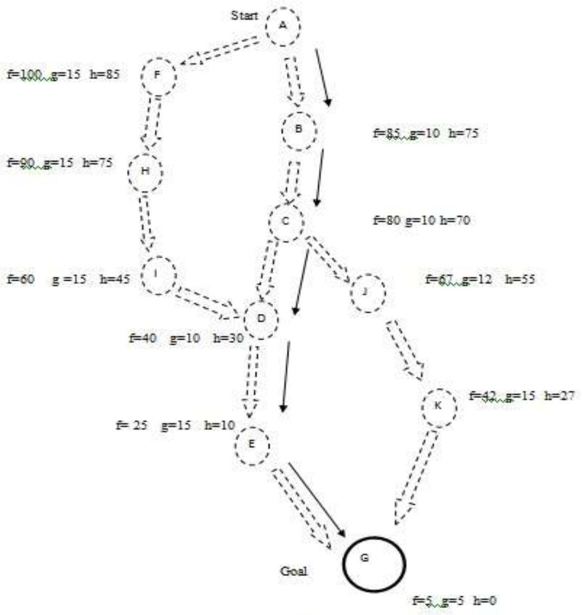

Figure 1

IV. Properties of A* search

We can highlight three properties of A* search, they are:

- Admissibility
- Consistency
- Dominance

⟹ **IV.I Admissibility** :

Let, n is our starting node and g is our goal node. The cost of moving from n to g is denoting by c (n, g).

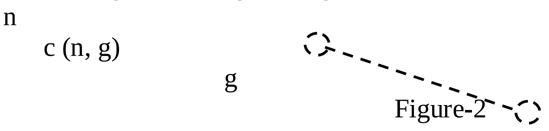

Figure-2

If the cost of moving from n to g node is zero, that is

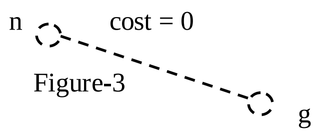

Figure-3

Then we find, $c(g) = h(n) + c(n,g)$

If h(n) is the heuristic of n, then heuristic value never overestimates path cost c (n, g).

$h(n) <= c(n, g)$

So, if h is admissible then A* search will find optimal solution. We can make it proof by contradiction.

Let, g is the optimal goal. g_2 is the suboptimal goal.

So, $c(g_2) > c(g)$

C* is the optimal cost. Now we assume that A* explore g_2 as goal.

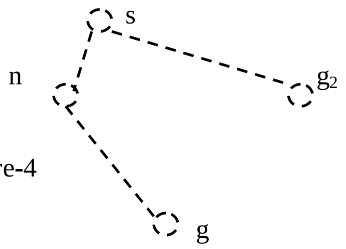

Figure-4

So, $g(g_2) + h(g_2) <= g(n) + h(n)$

$c(g_2) <= g(n) + h(n)$

$c(g_2) <= g(n) + cost(n,g) = c^*$

So, $c(g_2) <= c^*$

So, it is contradicted to our assumption. g_2 cannot be the optimal solution.

⟹ **IV.II Consistency**:

h will be consistent if

$\forall$ a,b $h(a) <= h(b) + cost(a,b)$

Example of consistency, from the below figure-5, we assume that,

h(a)=3 h (b) = 2 cost (a, b) = 1 g(b) = 4

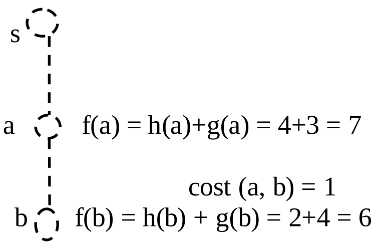

Figure-5

So, cost = h(b) + cost (a,b)
= 2 + 1
= 3

So, 3 <= 3 , then it is true.

⟹ **IV.III Dominance**:

h1 dominates h2 iff h1 (n) >= h2 (n) ;n

A* will explore a node n if,

$f(n) <= c^*$

⇨ $h(n) + g(n) <= c^*$

⇨ $h(n) <= c^* - g(n)$

Now, considering h1, h2

$h1(n) <= c^* - g(n)$

$h2(n) <= c^* - g(n)$

$h2(n) <= h1(n) <= c^* - g(n)$

Since, h1(n) >= h2(n) any node explore using h1 will be explored using h2.

So, h1 dominates h2.

V. Proof of completeness of A*

Here will be proved the completeness of the A* search. Before proving the completeness of A*, we have to prove that,
f- Value is non- decreasing along any path,

$f(a) = g(a) + h(a)$ [According to consistency property]

$f(a) <= g(a) + cost(a,b) + h(b)$

$f(a) < = g(b) + h(b) = f(b)$

$f(a) <= f(b)$

So, it is proved that f-value will be non-decreasing along any path.
Now we will prove that the completeness of A*.

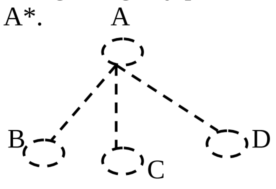

Figure-6

Let, A is the state that is removed from the queue. State of the fringe has higher or equal f-value than A
(As we have proved that f-value will be non-decreasing along any path.)
Let B, C, D are the states formed by exploring node A.

So, $f(B), f(C), f(D) >= f(A)$

Then, A* will explore next node whose f-value >= f(current)
So, A* always explore in non-decreasing order of f-value.
Finally, A* will ultimately explore the goal state.
So, A* is complete.

VI. Dark Side of the A* Search

Though A* search is one of the most efficient searching technique, but it has also some dark side. Such implementation of A search, required lots of memory. In general case its complexity is, O (number of the states) For a huge search application, it may occur run out of memory.

VII. Future work

By using the A* search technique's heuristic character, it may possible to develop the artificial intelligence sector of the computer science. Like, implementing this searching to some kind robots like, which are capable with some sensor devices like light, sound or radio wave to find the way to quick exit path. And this can be used to some rescue operation, in coal mine, determining slope from the surface.

VIII. Conclusion

The theme of this paper is to show the efficiency of A* searching technique over other searching techniques in computer science. From the above discussion and some simulations, it can be seen that A* search is complete, optimal, admissible and consistent heuristic function. In its operation, it will expand only the minimal number of nodes to get the goal. By this method it ensures the optimality. And if there is a goal is present in the graph, then obviously A*will find it.

References

Conference Papers:

[1] Rong Zhou and Eric A. Hansen, *Sweep* A*: Space-Efficient Heuristic Search in Partially Ordered Graphs *15th IEEE International Conference on Tools with Artificial Intelligence ² Sacramento, CA ² November 2003*

[2] R. Zhou and E. Hansen, Sparse-memory graph search Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2003), pages 1259–1266, 2003.

Journal Papers:

[3] T. Ikeda and H. Imai. Enhanced A* algorithms for multiple alignments: Optimal alignments for several sequences and k-opt approximate alignments for large cases. *Theoretical Computer Science*, 210(2):341–374, 1999.

[4] R. Korf. Linear-space best-first search. *Artificial Intelligence*, 62:41–78, 1993.

Book:

[5] Stuart Russell, Peter Norving "Artificial Intelligence A Modern Approach" *[Second Edition]*

Magazine:

[6] Bryan Stout, Smart Moves: Intelligent Path finding *Game Developer Magazine, July 1997*

Web Site:

[7] IIT KHARAGPUR NPTEL ONLINE

[8] Wikipedia *http://en.wikipedia.org/wiki/A*_search_algorithm*