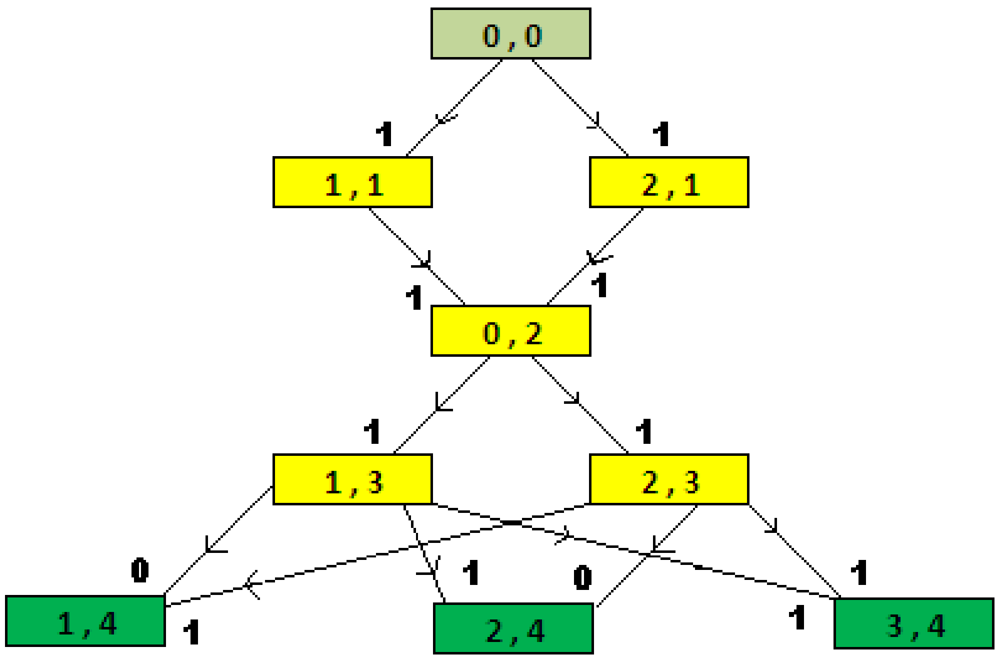
Heuristic function for finding the path using A star

Asked 12 years, 10 months ago Modified 12 years, 10 months ago

Viewed 21k times

5

I am trying to find a optimal solution for the following problem



1. The numbers denoted inside each node are represented as `(x,y)`.
2. The adjacent nodes to a node always have a `y` value that is (current nodes y value +1).
3. There is a cost of 1 for a change in the `x` value when we go from one node to its adjacent
4. There is no cost for going from node to its adjacent, if there is no change in the value of `x`.

5. No 2 nodes with the same `y` value are considered adjacent.

The optimal solution is the one with the lowest cost, I'm thinking of using A* path finding algorithm for finding an optimal solution.

My question, Is A* a good choice for the this kind of problems, or should i look at any other algorithm, and also i was thinking of using recursive method to calculate the Heuristic cost, but i get the feeling that it is not a good idea.

This is the example of how I'm thinking the heuristic function will be like this

- The heuristic weight of a node = Min(heuristic weight of it's child nodes)
- The same goes for the child nodes too.

But as far as my knowledge goes, heuristic is meant to be an approximate, so I think I'm going in the wrong direction as far as the heuristic function is concerned

algorithm graph-algorithm path-finding a-star heuristics

Share

Improve this question

Follow

asked Feb 4, 2012 at 12:28

Vamsi

4,253 ● 7 ● 51 ● 74

3 The heuristic *must* be admissible (or it won't be A* but just A), not just "approximate". It also should to be fast to compute, otherwise you lose more than you gain. And on top of all that, it should be "useful" - a constant zero heuristic would work if all costs are non-negative, but then A* degrades to BFS. – user555045 Feb 4, 2012 at 13:01

So what is your thought on my current heuristic – Vamsi Feb 4, 2012 at 13:38

Well, you can do it in O(n) (only incurred once), which isn't extremely bad, but it does mean touching every node (and remembering all results) so A* kind of loses its advantage. The point is, after all, *not* to explore the entire node-space (so it is *not* O(1) amortized). Also as written it's only ever 0 or 1, not exactly "useful", perhaps you missed a +1? – user555045 Feb 4, 2012 at 14:14

2 Answers

Sorted by: Highest score (default)

16

A* guarantees to find the lowest cost path in a graph with nonnegative edge path costs, provided that you use an appropriate heuristic. What makes a heuristic function appropriate?

First, it must be *admissible*, i. e. it should, for any node, produce either an underestimate or a correct estimate for the cost of the cheapest path from that node to any of goal nodes. This means the heuristic should never *overestimate* the cost to get from the node to the goal.

Note that if your heuristic computes the estimate cost of 0 for every node, then A* just turns into breadth-first exhaustive search. So h(n)=0 is still an admissible heuristic, only the worst possible one. So, of all admissible heuristics, the tighter one estimates the cost to the goal, the better it is.

Second, it must be cheap to compute. It should be certainly O(1), and should preferably look at the current node alone. Recursively evaluating the cost as you propose will make your search significantly slower, not faster!

The question of A* applicability is thus whether you can come up with a reasonably good heuristic. From your problem description, it is not clear whether you can easily come up with one.

Depending on the problem domain, A* may be very useful if requirements are relaxed. If heuristic becomes inadmissible, then you lose the guarantee of finding the best path. Depending on the degree of overestimation of the distance, hovewer, the solution might still be good enough (for problem specific definition of "good enough"). The advantage is that sometimes you can compute that "good enough" path much faster. In some cases, probabilistic estimate of heuristics works good (it can have additional constraints on it to stay in the admissible range).

So, in general, you have breadth-first search for tractable problems, next faster you have A* for tractable problems with admissible heuristic. If your problem is intractable for breadth-first exhaustive search and does not admit a heuristic, then your only option is to settle for a "good enough" suboptimal solution. Again, A* may still work with inadmissible heuristic here, or you should look at beam search varieties. The difference is that beam searches have a limit on the *number* of ways the graph is currently being explored, while A* limits them indirectly by choosing some subset of less costly ones. There are practical cases not solvable by A* even with relaxed admissbility, when difference in cost among different search path is slight. Beam search with its hard limit on the number of paths works more efficiently in such problems.

Share Improve this answer Follow

answered Feb 4, 2012 at 13:52

kkm mistrusts SE
5,490 ● 2 ● 35 ● 64

2

Seems like an overkill to use A* when something like Dijkstra's would work. Dijkstra's will only work on non-negative cost transitions which seems true in this example. If you can have negative cost transitions then Bellman-Ford should also work.

Share Improve this answer Follow

answered Feb 4, 2012 at 14:38

Justin
4,216 ● 4 ● 27 ● 49