Final Solution

CS510, Fall 2012, Drexel University

Ariel Stolerman

1)

Grade: 5/5

Question:

There is a refinement of the A^* algorithm called the A^*_ε algorithm. Assume that you are given a heuristic $h_F(n)$ that estimates the remaining computational effort required to find a goal state from node n. Rather than ordering the fringe according to ascending $f(n)$ costs, A^*_ε orders the fringe according to ascending $g(n) + h_F(n)$ values. You may assume that $h_F(n)$ never overestimates the amount of remaining computational effort and obeys the triangle inequality.

Is A^*_ε guaranteed optimal? Why or why not?

Answer:

Optimality is measured with respect to some cost determined for the path (the terms g looks at), not necessarily the computational effort invested in the process. For instance, when calculating the path from Arad to Bucharest, we shouldn't care (within reason) about the computational cost (given it is feasible) of calculating the path, but we should care about achieving the shortest path possible (in kilometers / time). However, when sorting the fringe w.r.t. $g(n) + h_F(n)$, we may not get the shortest path. For instance, assuming each operation on some integer constant is the same (reasonable assumption, given all integers, small to large, take 32 bits anyway), and given we're at some node on our way to the path, consider the following 2 alternative paths to the goal:

1. A path with edge cost of 1 (not computational cost, real path cost), that includes 10 edges.
2. A path with edge cost of 100, that includes only one edge.

It's reasonable to assume that the calculation of path (1) will necessitate more resources than that of path (2); however it's easy to see that in fact path (1) is optimal. Therefore using A^*_ε will result with suboptimal solution, choosing the path that runs through (2) instead of through (1). Therefore A^*_ε does NOT guarantee optimality, as defined above w.r.t. the solution (and not computational power required to calculate it).

Note: although $h_F(n)$ is consistent (triangle inequality) and admissible w.r.t computational power, it doesn't matter. In fact, since it is defined as such, we know for sure that in the example above path (2) will most-probably be chosen instead of (1).

2)

Grade: 5/5

Question:

Under what circumstances might you want to use A^*_ε over regular A^*?

Answer:

I would ALLOW usage of A^*_ε where the limitation described in the previous question will not hold, and that would be when the real action costs are fairly equal, e.g. road segments in Hungary are about the same length. This eliminates as much as possible the effect described previously, since it brings the ranking of a reasonable heuristic and h_F closer together.

I will WANT to use A^*_ε when the problem necessitates long calculation time and I have limited resources, i.e. when processing time is of high priority.

For instance, path planning in GPS systems with ever changing terrain (if such scenario is even possible) - path recalculations are constantly processed, time is of the essence, and there are many paths to choose. Here I might allow losing path efficiency in favor of more real-time calculation (I don't think that's a realistic scenario with today's technology, but nonetheless an example).

3)

Grade: 5/5

Question:

Let us consider the special case when $h_F(n)$ always returns a constant value; in such an instance, like what other algorithm will A^*_ε behave? Why?

Answer:

In case $h_F(n)$ always returns a constant value, the A^*_ε algorithm becomes a sort of Uniform-Cost Search: Unifrom cost search sorts the fringe by $g(n)$ values, and when $h_F(n)$ always returns some constant C, sorting by $g(n)$ values is equivallent to sorting by $g(n) + C$ values. Therefore A^*_ε will behave exactly like Unifrom cost search.

4)

Grade: 10/10

Question:

Let's assume – in addition to $h_F(n)$ – you know a "traditional" A^* heuristic, $h(n)$, that estimates the remaining path-cost to the solution. $h(n)$ is admissible. Also, you know that:

$$\forall n: h_F(n) \leq h(n) + k$$

where k is some known constant. With this knowledge, is it possible to modify A^*_ε such that it is optimal for arbitrary problems? If not, why? If so, how?

Answer:

Yes, it is possible under one assumption mentioned later. Since $h(n)$ is admissible w.r.t. the value we care about – path cost – and we know that for every node n: $h_F(n) \leq h(n) + k$, then we know that: $h_F(n) - k \leq h(n)$, therefore $h_F(n) - k$ will never overestimate w.r.t the path cost (as well as w.r.t. the computational effort, as before, given $k \geq 0$). This makes $h_F(n) - k$ admissible (again, w.r.t. the path cost, just like $h(n)$ is admissible).

We also know that $h_F(n)$ holds to the triangle inequality, i.e. for given node n, successor n':

$$h_F(n) \leq cost(n, n') + h\_F(n')$$

Therefore the following also holds:

$$h_F(n) - k \leq cost(n, n') + h_F(n') - k$$

Which makes $h_F(n) - k$ also consistent. Here comes in the assumption we need: I assume the cost function for $h_F(n)$ should refer to the real computational effort cost, not the real path cost; in that case we need the assumption that for every path $n \to n'$:

$$comp - effort - cost(n, n') \leq path - cost(n, n')$$

This way we are guaranteed that the consistency derived above is indeed correct w.r.t path cost.

So under the assumption above, $h_F(n) - k$ is admissible and in fact consistent w.r.t path cost, therefore we need to change A^*_ε to use$h_F(n) - k$ instead of $h_F(n)$ to secure optimality for arbitrary problems.

5)

Grade: 2/2

Question:

For this and the subsequent three problems, consider the following language:

- $Student(s)$: s is a student
- $Class(s, c)$: s is a student in class c
- $Grade(s, c, g)$: s earned the grade g in class c

And the following sentence in this language:

$$\forall s_1 \forall s_2 Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A) \Rightarrow s_1 = s_2$$

Is this sentence represented in Propositional Logic? Why or why not?

Answer:

The sentence above is NOT represented in propositional logic since it contains quantifiers, which are not part of propositional logic. It is in fact represented in first-order logic.

6)

Grade: 4/4

Question:

What is the meaning of this sentence in plain English?

Answer:

Hoping it is not a forecast, the sentence means:

“There can be only one student in class CS510 that earns a grade A.”

Note: but it doesn't mean that such student indeed exists.

7)

Grade: 4/4

Question:

What would be the meaning of this sentence if the implication operator (⇒) were replaced by a conjunction (∧)?

Answer:

I think the meaning would change to:

"There is only one student in class CS510, and his grade is A."

Assuming that there exists at least one student in the world, otherwise it should be: "Assuming the world is not empty, there exists only one student in class CS510, and his grade is A."

However, more literally, this sentence says: "for every 2 objects s1 and s2 in the world: they are students, equal, and have the grade A in class CS510"

But the world contains also non-student objects, (like class CS510), so something is problematic with the logic of the sentence (assuming classes can't be students). It would have been resolved if the sentence would refer only to the students in the world:

$$\forall s_1 \forall s_2 Student(s_1) \wedge Student(s_2) \Rightarrow Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A) \wedge s_1 = s_2$$

But perhaps I overanalyzed this for a 4-point question...

8)

Grade: 5/5

Question:

Without modifying the language, rewrite the original sentence such that it does not use universal quantification. (Hint: You *may* use other types of quantification.)

Answer:

$$\forall s_1 \forall s_2 Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A) \Rightarrow s_1 = s_2$$

is equivalent to:

$$\neg\neg \forall s_1 \forall s_2 Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A) \Rightarrow s_1 = s_2$$

Using DeMorgan rules:

$$\neg \exists s_1 \exists s_2 \neg [Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A) \Rightarrow s_1 = s_2]$$

Where:

$$Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A) \Rightarrow s_1 = s_2$$

Is equivalent to:

$$\neg [Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A)] \vee s_1 = s_2$$

So the original sentence is equivalent to:

$$\neg \exists s_1 \exists s_2 \neg [\neg \{Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A)\} \vee s_1 = s_2]$$

Which is equivalent to (the final answer):

$$\boxed{\neg \exists s_1 \exists s_2 Student(s_1) \wedge Student(s_2) \wedge Grade(s_1, CS510, A) \wedge Grade(s_2, CS510, A) \wedge \neg (s_1 = s_2)}$$

Which in plain English means:

"There does not exist a pair of students that both have a grade A in class CS510, and they are not the same person."

9)

Grade: 5/5

Question:

Consider the perennial Festivus Candy problem: There is a child in a room with some candy that is hanging – out of reach – on the Festivus pole. A stool is available that will enable the child to reach the candy if he or she climbs on top of it. Initially, the child is at location A, the candy at location B, and the stool at location C. The child and stool have hight low, but if the child climbs onto the stool he or she will have height high, which is the same height as the candy. The actions available to the child include GOing from one location to another, PUSHing an object from one place to another, CLIMBing onto an object, and GRASPing an object. Grasping an object results in holding the object if the child and object are both in the same place and at the same height. The goal, of course, is for the child to be holding the candy. Here are the actions in STRIPS notation:

Action($Go(f,t)$,

Precond: $At(Child,f)$

Effect: $\neg At(Child,f) \wedge At(Child,t)$)

Action($Push(o,f,t)$,

Precond: $At(Child,f) \wedge At(o,f) \wedge Height(Child,low)$

Effect: $\neg At(Child,f) \wedge At(Child,t) \wedge \neg At(o,f) \wedge At(o,t)$)

Action($Climb(o)$,

Precond: $At(Child,\ l) \wedge At(o,l) \wedge Height(Child,low)$

Effect: $\neg Height(Child,\ low) \wedge Height(Child,high)$)

Action($Grasp(o)$,

Precond: $At(Child,\ l) \wedge At(o,l) \wedge Height(Child,h) \wedge Height(o,h)$

Effect: $Holding(Child,o)$)

What are is the initial state and goal state for this problem instance?

Answer:

The states below exclude checks that $Child$ is a child, A is a location etc. (as do the preconds in the actions defined above):

Initial state:

$$At(Child,A) \wedge Height(Child,low) \wedge At(Candy,B) \wedge Height(Candy,high) \wedge At(Stool,C) \wedge Height(Stool,low)$$

Goal state:

$$Holding(Child,Candy)$$

(There's no $\neg Holding(Child, Candy)$ in the initial state because I assume this illogical scenario, given the Child and Candy's separate positions and heights, cannot happen)

10)

Grade: 5/5

Question:

What benefit, if any, would there be in solving this problem using Partial Order Planning?

Answer:

Partial order planning in this case cannot help by allowing efficiency derived from the ordering being partial, since the problem defined above contains a very strict order of operations that has to be done linearly, and no parallelism or partial order can really exists (i.e. the only reasonable plan for the problem above is totally ordered):

- In order for the child to get the candy, he has to be high in B
- In order for him to be high in B he has to be on a stool in B
- In order for him to be on a stool in B he has to climb a stool in B

... (backward chaining in plain English). There is no degree of freedom left in this problem (other than illogical operations like going back and forth from A to B for instance).

The only benefit I can think of is using graph plan which may be efficient for solving the problem, however the resulted plan would have a total order.

11)

Grade: 5/5

Question:

In general, what is the primary disadvantage of state space planning (i.e., planning using search methods like A^*)?

Answer:

The primary disadvantage of using state space planning - using search methods - is that the search can be overwhelmed by irrelevant actions, i.e. the branching factor can be very large with many redundant actions at every level, and the search will become infeasible.

12)

Grade: 5/5

Question:

What if there are multiple Festivus polls in the room, only one of which has candy? You cannot be sure whether or not a given poll has candy until you try and GRASP it. In other words, the GRASP action is now non-deterministic: it has a disjunctive set of effects, including a new "$\neg Holding(Child, o)$" proposition. What techniques might you use to solve this modified version of the problem?

Answer:

To solve the problem described above, where we have incomplete information (whether some poll holds the candy or not), we can use conditional planning.

We will need to add an observation action $CheckHasCandy(x)$ with the effect $KnowsIf\left(\neg Holding(Child, Candy)\right)$, and use it after every GRASP action to see whether we can terminate (the "then" part - reached the goal) or continue the search for Candy (the "else" part - continue to another poll). In this case I would also add a predicate to indicate whether some poll has been visited.

Another technique we can use is monitoring and replanning, where we monitor whether the preconditions of the remaining plan/next action (execution/action monitoring, respectively) are met and replan accordingly. In this case it would be to check whether the Candy has been grasped. If not, we can go back and try another poll and so on until the Candy is reached.

13) *Extra Credit

Grade: 4/4

Question:

The terms *"autological"* and *"heterological"* have the following meanings:

- An adjective is *autological* if and only if it describes itself. For example, "short" is autological because the word "short" is short. Other examples of autological adjectives include "English," "polysyllabic," and "sophisticated."
- An adjective is *heterological* if and only if it does not describe itself. For example, "long" is heterological because the word "long" is not long. Other examples of heterological adjectives include "monosyllabic" and "pulchritudinous."

Is "heterological" heterological? Is it possible to express this concept in first order logic? If so, how? If not, why?

Answer:

The word "heterological" is heterological iff it does not describe itself. However, it is known that any good definition of a term must NOT use the term itself, therefore the word "heterological" infact does NOT describe itself, so it IS heterological.

I believe this concept can be expressed in first order logic under some representation assumptions. Relations are simply sets of tuples, so a relation can be an argument in a tuple contained in that same relation (contains itself recursively). So if you indeed allow addressing relations as both relations and constants, you can say:

$$\forall x: \neg x(x) \Leftrightarrow heterological(x)$$

However, I think this ambiguity of context may not be allowed in first order logic, in which case the answer would be no. So my final answer is: No, I don't believe it can.