**Artificial Intelligence - Assignment 1**

**Question 1 - c**

**Sameer Anees Jaliawala**

**Sj02732**

**d) Why is it important to have an admissible heuristic in A\* to ensure Optimality?**

ANS. A heuristic function is said to be admissible if it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path (*Source: Wikipedia*). The search algorithm uses the admissible heuristic to find an estimated optimal path to the goal state from the current node. We use the following function to evaluate A\*: *f(n)=g(n)+h(n),* where h(n) is the estimated cost to goal from current state and g(n) is the cost till the current state. For an admissible heuristic, we need: 0 ≤ h(n) ≤ h\*(n), (h\*(n) is the actual cost to reach a goal). If an admissible heuristic is used, then A\* would eventually find an optimal solution to the goal state by lowering the quantity f(n). The algorithm will expand all those nodes with the lowest f(n), which recommends h(n) to be less than the estimated cost to goal node. For example, if h(n)=0, then the algorithm will guarantee the shortest path and might act like Dijkstra’s algorithm, or if h(n)=h\*(n), then the algorithm will follow the most ideal path. If an admissible heuristic is not used, then we can have an estimation that is bigger than h\*(n). The algorithm will not traverse the least cost path if the high estimation ends up at this path; instead it will try to find another path, which will not actually be the least cost path.

One of the reasons why A\* algorithm can handle admissible heuristic to provide optimality is because it sorts all nodes in ascending order of f(n), which allows us to traverse the most promising frontier first. It also stops when it expands the goal node and searches all the optimal paths rather than just provoking the goal node for the first time. That is why A\* search can handle both over-estimate and admissible heuristics in its algorithm; however, it will only guarantee optimality if an admissible heuristic is used.

**e) Several enhancements have been proposed to A\* algorithm. Discuss some variant of A\* and its motivation and working. Give references for the technique discussed.**

ANS. One of the enhancements that has been proposed is to include multiple goals in the search algorithm. This was proposed by Dmitry Davidov and Shaul Markovitch in their paper called “Multiple-Goal Heuristic Search”. Their idea is to achieve as many goals as possible within the allocated resources. One of the reasons why this enhancement was proposed was due to the success criterion difference between heuristic search and focused crawling, while attempting to search large portions of the web. It is stated that the heuristic search is completed as soon as a single goal state is found, while focused crawling continues to reach as many goal states as possible. However, changing the success criterion of the heuristic search is not enough; the need to change the heuristic function is even more important to achieve multiple goal searches. Some examples of where we might need multiple goals are:

* Genetic Engineering, to find multiple possible alignments of DNA sequences.
* Chemistry, to find multiple substructures of a complex molecule.
* Robotics, to plan paths for multiple robots to access multiple objects.

In some of these cases, we could invoke single-goal search multiple times. However, it might be wasteful, and we may wish to exploit the multiple-goal nature of the problem to make the search more efficient, in resource-bounded computations.

The framework that has been identified by the authors for the multiple heuristic search problem is:

1. Input:

• A set of initial states *Si ⊆ S*

• A successor function Succ : *S → 2s* such that *Succ(s) = {s′ | <s, s’> ∈ E*}. (Sometimes Succ is given implicitly be a finite set of operators O.)

• A goal predicate *G : S → {0, 1*}. We denote *Sg = {s ∈ S | G(s)*}. Sometimes Sg is given explicitly.

1. Search Objective: Find a set of goal states *SRg* ⊆ *Sg* which satisfies:

• For each *s ∈ SRg*, there is a directed path in < S,E > from some si ∈ Si to s.

• The search resources consumed should not exceed R.

1. Performance evaluation: | *SRg |*. Obviously, higher values are considered better.

The authors then make their way to A\* search algorithm where they explain how the original algorithm can be converted to handle multiple goal search by collecting each goal it finds in its focal list. These are goals with e−optimal paths. We can use multiple-goal heuristic to select a node from the focal list for expansion. The algorithm stops when all the allocated resources are exhausted or when all the nodes in focal satisfy f(n) > (1+ǫ) gmin, where gmin is the minimal g value among the collected goals. Multiple-goal backtracking works similarly to the single goal version. However, when a goal is encountered, the algorithm simulates failure and therefore continues. A multiple goal heuristic can be used for ordering the operators in each node.

The authors also define some heuristics that we can use to implement in multiple goal A\* search. Two of them are mentioned below:

1. The Perfect Heuristic: Assume that the given resource limit allows us to expand additional M nodes and look at all the frontiers of size M rooted at the current list of open nodes. A perfect heuristic will select a node belonging to the frontier with the largest number of goals.
2. Marginal – Utility Heuristic: This heuristic not only considers the expected benefit of the search but also the expected cost. We prefer subgraphs where the cost is low, and the benefit is high. Hence, this heuristic would try to estimate a high return for our resource investment. (Davidov and Markovitch 2006)[[1]](#footnote-1)

1. Davidov, Dmitry, and Shaul Markovitch. 2006. "Multiple-Goal Heuristic Search." *Journal of Artificial Intelligence Research* (26): 417-425. [↑](#footnote-ref-1)