Summary of Dynamic Repairing A* Algorithm

Saksham Goel

September 26, 2018

Abstract

Dynamic Repairing A* also referred to as DRA* algorithm is a proposed algorithm to tackle the problems of re-planning. The author suggest that in the context of re-planning, DRA* algorithm performs better with the situation of change in goal state or action weights by using the already computed states as compared to A* which would use a strategy of computing the answer from scratch. This assignment is used to summarize the paper and answer three critical questions.

1 Summary

The problem of re-planning is a very widely known problem, which arises because of change in a goal state of a plan or action weights of the plan. The problem of re-planning has always been tried to be solved in two ways:

- **Planning from Scratch** refers to the technique of not using any knowledge from the previous plan even if it some progress was made. Need to start making a new plan which should be implemented to achieve the goal based on the new changes.
- **Plan Repairing** refers to the techniques which uses the knowledge from previous executed plan (some progress made) to redefine the next steps to achieve the goal with the new changes

The author recommends a new approach called Dynamic Repairing A* to tackle this problem of re-planning based off the Plan Repairing Approach. DRA* uses A* search algorithm and modifies it to make sure to extract as much information from the progress made in the previous plan and then use this information to compute the next steps considering the new changes. The most import difference between DRA* and A* is the concept of handling predecessor states and the concept of informing and valid states. DRA* makes sure to compute the new g-values for each and every predecessor from the new state that can be computed to make sure that the new g-values reflect the changes made in the plan and then uses A* on top of these updated scores while making sure to account for states that have been already visited earlier (original plan progress) to find the new way to goal. Other than this, it also includes some other ways of reducing the time of computation for some special cases so that the answer does not actually require running the bulk of the algorithm.

2 Dynamic Terrain Useful

According to me, DRA* algorithm is very useful in a dynamic terrain where the action weights can change because of the change in environment/terrain. I think this algorithm is very useful in the field of Navigation like Google Maps. Navigation is a dynamic terrain because there are many other entities each acting independently and continuously moving thus creating different scenarios. Considering a problem of reaching from Point A to Point B when driving in a car. Finding a route which is the fastest is what makes Google Maps very useful, however traffic is a dynamic entity, hence because of jams/congestion/road closures/accidents there can be a change in the time taken along some route. This can be attributed to like change of the weights of the action in a plan.

In this situation starting position was Point A, goal was Point B and the plan was to reach from Point A to Point B in the least time possible, however because of traffic jam along a route, the time taken along that road has suddenly increased as compared to original time that was computed. This calls for re-planning. Instead of using a planning from scratch, it is better to repair the original plan, because already some progress has been made. This is where DRA* fits perfectly well, because it can use the computations from the originally generated plan/graph to compute the new route to reach Point B from the current point (some detour from the current location without any traffic) considering the new information about the time taken along routes. I think Google Maps, Apple Maps etc can use or are already using this kind of search to save a lot of time.

3 Dynamic Terrain not Useful

For me there are two different scenario where this type of algorithm might fail.

The first one is when the goals are meant for short time in a long term plan and they keep changing, like in a game as Chess. In Chess although the long term goal is to Win but there will be short term changing goals (based on strategies) to achieve a particular state of advantage. In this kind of scenario, because goals can keep on changing and be very well very different from the original goal, it is better to reevaluate the whole plan before the next move as compared to using the knowledge from the previous moves/plan. Consider an example, where a person starts a strategy to try to get the opponents rook in a game of chess. However in the sequence of moves, the opponent is able to get the players bishop or queen or something else, which will make the player to move from an attacking play to a defensive play. Going from attacking to defending is something that requires thinking of the strategy from scratch. This is where it is important to consider using a simple planning from scratch for re-planning to have better chances of winning.

The other type of scenario is something like constructing a very critical machine (like a satellite or rocket) which cannot be repaired again and again and are very safety critical. In these types of scenario if the goal of the machine

changes, it is better to re-plan the whole thing as compared to build upon the same machine and try to make it work for the new goal setting.

4 Performance Measure

DRA* algorithm is very different from other mentioned state-of-the-art algorithms because it can cope with both changes in goal set and action costs. Other than that, as compared to some other state of the art algorithms like GAA*, MPGAA* which can handle both changes using techniques to generalize a better heuristic value accounting for overall moving targets and changing action costs or running multiple times to save the cost to other possible states, DRA* uses a better technique of actually realizing the f-values of other states based on the progress made in the original plan, and use these f-values to actually compute the best next steps based on the change. This makes sure that DRA* is not run multiple times and also make sure that DRA* is not assuming or generalizing values but actually computing the best possible ways by accounting the changes in the already computed plan space/graph. However thinking of conditions where DRA* approach is not the most optimal one, is like when trying to catch some moving target in a simple planar space. Like designing a robot cat that tries to catch a mice in a house setting. Here as soon as the mice moves, it is better to use generalized f-values to direct the new approach/steps rather than use a plan repairing algorithm because dont need to spend time again calculating the new f-values. This is the case in this scenario because here the main goal is to catch the mice rather than minimizing the time. Because of this kind of goal, the earlier algorithms can use different strategy based movement to try to trap a mouse, which will be computationally expensive in case of DRA* because it will try to make a trap work by making sure that original did not really go to waste (process of informing).

In comparison to algorithms like A*, I think according to their evaluation section, Author mentions that the DRA* algorithm is better than A*, however only certain given conditions. The performance of DRA* with respect to A* greatly depends on the given graph structure (like branching factor) and also the amount of change in the given re-plan. As mentioned in the results section, the optimal performance of the DRA* algorithm is much better than A* only if the original plan has not been executed more than 50% and the change in goal set is not more than 20 - 50%. All of these reasons can be easily mapped to the functioning of DRA* algorithm, specifically the part of Fully and Lazy informing, which is directly affected by branching factor and other stuff.

So in the end, I would not necessarily say that DRA* outperforms the other state of the art algorithm however it certainly performs better than most of them in more generalized conditions.