Path Replanning Algorithms

Path planning\Path finding algorithms development started in the 1950’s with the Dijkstra’s algorithm. Since then a lot of progress has been made considering the complexities of real environment. Starting from uninformed search to informed search and dynamic continuously changing environment, a number of algorithms have been developed.

For informed search, A\* search has been the main algorithm. A\* search is a static algorithm in which the costs(heuristic) do not change with time. To handle dynamic environment, LPA\* (Lifelong Planning A\*), algorithm was developed by Sven Koenig and Maxim Likhachev. LPA\* is an incremental version of A\*. It applies to finite graph search problems on known graphs whose edge costs increase or decrease over time [Ref:3]

A variant of dynamic algorithm D\* ( Dynamic A\*) was developed by Anthony Stentz to consider dynamic changing environment in real time. D\* algorithm is capable of planning paths in unknown, partially known, and changing environments in an efficient, optimal, and complete manner. D\* is a planning algorithm that produces an initial plan based on known and assumed information, and then incrementally repairs the plan as new information is discovered about the world. Experimental tests showed that D\* is several hundred times faster than replanning from scratch for large problems. [Ref : 1, Ref : 2]

An easier to understand variant of D\*, D\* Lite was developed by Sven Koenig and Maxim Likhachev. It is similar in behavior to D\* but algorithmically different and simpler. It is based on LPA\* algorithm, only difference is that D\* Lite started search backward from goal to start node.

Results :

LPA\* vs A\* : LPA\* is better to A\* in terms of : (1) Fewer vertices expansion in LPA\*, (2) LPA\* is efficient because it performs heuristic searches and thus calculates only the g-values of those vertices that are important to determine a shortest path.

D\* Lite vs LPA\* : LPA\* works from start vertex to goal vertex. D\* Lite works in reverse direction – from goal vertex to start vertex. This showed a further improvement in performance for D\* Lite.

D\* vs D\*Lite – Experimental properties showed that D\* Lite is at least as efficient as D\*[Ref:3]. [Ref 4] shows that D\* Lite is significantly better in some cases. D\* Lite is simpler and easier to understand.

D\* vs A\* : Focussed Dynamic A\* (D\*) achieved a speedup of one to two orders of magnitudes over repeated A\* searches. [ Ref :3, Ref : 4]

Ref:

1. <http://www.ai.sri.com/~konolige/cs225b/stentz94optimal.pdf>
2. <http://idm-lab.org/bib/abstracts/papers/aaai02b.pdf>
3. <http://www.seas.upenn.edu/~maximl/files/tutorials/tutorial_icra10_v26.pdf>

( performance comparison – Slide 237,239)

1. [https://en.wikipedia.org/wiki/D\*](https://en.wikipedia.org/wiki/D*)
2. <http://www.frc.ri.cmu.edu/~axs/dynamic_plan.html>