Lifelong Dynamic Timed A* (LTA*) for Fastest Path Retrieval in Congested Road Networks Using Predicted Speeds

Kartikey Sondhi, Poulami Dalapati[a] and Saurabh Kumar[b]
Department of Computer Science and Engineering, The LNM Institute of Information Technology Jaipur, India

Keywords: Intelligent Transportation Systems, Traffic Congestion Management, Traffic Modelling and Simulation, Dynamic A*.

Abstract: Efficient transportation systems are crucial for the ever-growing smart cities. With the increasing urbanization and growth in vehicular traffic, congestion has become a significant challenge. This research paper addresses the critical issue of identifying the fastest, least congested path in road transport networks, aiming to enhance overall travel efficiency and reduce the negative impact of traffic congestion. The study employs an improved version of the Lifelong Planning A* (LPA*) that helps find the fastest route between two points in dynamic changing environments. The proposed methodology is called the Lifelong Dynamic Timed $A^*(LTA^*)$ algorithm with an optimal bound weight factor integrated with it to make the search more guided and efficiently predict optimized traffic paths to provide real-time recommendations. To validate the effectiveness of the developed algorithm, extensive simulations and case studies are conducted on a small area in Washington as well as on Grid Worlds. The experimental results show that LTA*, within accurate weight bounds, always managed to find the fastest path, and in some cases, the time taken was close to half of that produced by A*.

1 INTRODUCTION

Heuristic based searches have been widely used to compute the shortest paths in graphs, which extends is usage to grid world planning problems, mazes and even road networks. To find solutions to various problems algorithms are frequently used to add robustness to the the search. The search algorithms in the literature often give results with different costs, where costs are often measured in time to generate first solutions, deviation from ideal outcomes, computational units exhausted as well as total nodes expanded to find these solutions.

The A* algorithm being a special case of the Breadth First Search is often deployed in such problems, first proposed in (Hart et al., 1968), utilizes the heuristic to perform a guided search as compared to the unguided alternative being the Djikstra's algorithm, which also is a special case of the BFS. The conditions that the these algorithms will find the optimal solution if any, are that, firstly the heuristic function is admissible, meaning it never over-estimates the cost to the goal and the secondly, the edges within the graph are non negative. In 1970, Pohl proposed the inclusion of a weighted constant *w*, when multiplied to the heuristic in A* would speed up search by compromising with the optimal solution and receiving a sub-optimal one(Pohl, 1970), when *w* was greater than 1. This sub-optimal nature of the solution is bounded by the weight w, in cases where *h* is admissible , meaning the solution produced is to be not more than *w* times the optimal solution(Likhachev et al., 2005). This inflated heuristic search paved way for a new class of searching algorithms called the Anytime Algorithms. In these algorithms, a inflated value of w is chosen, that allows the algorithm to generate a first sub-optimal solution very quickly, and refines over its search by iterating over smaller values of *w* to produce subsequent better solutions as the search progresses. The way these first values of *w* are chosen and the magnitude of their decrements is something that varies greatly from one problem to the other, but someone interested in reading can refer to (Richter et al., 2010)(Stern et al., 2014)(Van Den Berg et al., 2011). When speaking about urban transportation systems the actions taken now, greatly reflect the next possible choices we can possibly make. For these very reasons we ponder through the paper the reasons and lengths to which the weighted constant has been

[a] https://orcid.org/0000-0003-0539-3029
[b] https://orcid.org/0000-0002-1786-1056

Sondhi, K., Dalapati, P. and Kumar, S.
Lifelong Dynamic Timed A* (LTA*) for Fastest Path Retrieval in Congested Road Networks Using Predicted Speeds.
DOI: 10.5220/0012629900003702
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 10th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2024)*, pages 293-298
ISBN: 978-989-758-703-0; ISSN: 2184-495X
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

applied.

These anytime algorithms designed are great for generating the quickest first solutions but often fail to perform well in dynamic environments like transportation systems and robot path finding, where the edge costs are prone to change. The D* (Dynamic A*) (Likhachev et al., 2005), D* Lite and LPA* (Lifelong Planning A*) (Koenig et al., 2004) are dynamic versions that are adapted to the changing edge cost that may occur in the environment. They compute a first shortest path and then based on the f values of remaining states, calculates alternate optimal paths when current optimal trajectories get obstructed. The difference between the D* Lite and LPA* algorithms is the cost associated with computing the solution in real time, with the D* Lite being the optimised and refined version, with significantly lesser push and pops from the priority queue and the restarting nature of the LPA*.

To extend the knowledge of such algorithms into Intelligent Transportation System (ITS), we must address that there are critical challenges to ITS like fluctuations in road mobility volumes, sensitive to irregular patterns and real time traffic control, which, due to the highly stochastic nature of congestion on road networks, creates planning problems. Path finding in the case of vehicles not only requires shortest routes to the destination, but a path that avoids excessive congestion, includes better quality roads and exhibit better or even the fastest travel times possible. To tackle these spatio-temporal factors, congestion data needs to be captured from road networks and processed. For this, in (Ma et al., 2015), Ma et al propose a Long Short Term Memory Neural Network that evaluates traffic data collected through microwave sensors, that produced speed predictions better than Support Vector Machines and Non-Linear Auto-Regressive Neural Networks, with the model proposed delivering speed prediction with less than 4% Mean Absolute Percentage Error(MAPE). Another such state of the art method was proposed by Zhang et al in 2020 in (Zhang et al., 2020) where the traffic was predicted using a Structured Learning Convolutional Neural Networks. Reflecting upon the possibilities of NN's to assist in the prediction problem of this highly dynamic nature of traffic on road networks, a novel Lifelong Planning Timed A* (LTA*) is proposed, that shall produce fastest paths in real time, even in cases of accidents or abrupt road closures.

The remaining paper is organized as follows: Section 2 proposes the novel LTA* with the mathematical model and the algorithmic approach. The theoretical analysis with challenges and the experimental results and comparative analysis have been discussed in Section 3. Finally Section 4 concludes the proposed research work and discusses the future scope.

2 METHODOLOGY

2.1 Problem Statement

The goal of this paper is to propose and develop an algorithm that suggests the fastest route to be taken between two points on a real world road network map. Our proposed algorithm encapsulates the challenges faced in real road networks, like the unreliability of congestion that may be present on roads, and it modifies itself in times of uncertain traffic conditions in real time.

The problem involves receiving a map of city, that can be represented as a graph with the vertices represent intersections and edges represent the roads connecting them. If a person wants to travel from one intersection to the other, the algorithms returns the path, which takes the shortest time to travel. The algorithms uses historical speed data and predictive models, developed for certain cities using sensors and machine learning in real time to retrieve the fastest path.

2.2 Proposed LTA*

In this section we will introduce the working of our LTA* algorithm, which tackles the problem of the stochastic nature of congestion that might be present on road networks, then deriving a shortest time consuming path from within such a network with great accuracy. Doing so requires

- A time dependent value for each edge, containing values for every 10 minute time interval or real time generated values. These values need to be updated using the sensors in the environment or through online maps that show road closures.
- A hash map is used that stores the speed data for the different times of the day, and is only updated in cases of significant difference between predicted values or sudden road closures.
- Further the LTA* algorithm proposed is inspired from the Lifelong Planning Problem in (Koenig et al., 2004). The given algorithm uses an additional variable for each node and that is it's t also called the t-score, a parallel drawn from the *g_score* and *f_score*, that stores the time at which a certain node can be reached at.
- The g also called *g_score*, for a node u, is updated based on the least time it takes to get to u from

its predecessors. Intuitively doing so would make the dimensions of this physical quantity to that of T^1. To change the dimensions back to those of distance, we multiply the *MaxSpeed* of the entire network so that we are logically diving the distance between two intersections by a quantity between $(0,1]$ and then updating the g.

Assuming we have to update the g value for some state s which has a predecessor s', so in equation 1.

$$\begin{cases} g(s';u') = g(s) + \frac{c(s,s')*MaxSpeed}{Speed(s',s,t(s'))} \\ f(s') = g(s') + w*h(s') \end{cases} \tag{1}$$

Algorithm 1: *LTA** Main().

```
procedure Main( )
    Initialize
    Repeat                                          ▷ forever
        location ← getLocation()
        currTime ← getTime()
        ComputeFastestPath()
        wait for change in predicted values
        for all updated edges from (u,v)
        UpdateNode(v)
end procedure
```

The working of the algorithm is similar to the optimised LPA* proposed in (Koenig et al., 2004), but the overview of the algorithm is as follows:

- The edge costs are updated as per the normalised distance from t values of the predecessors instead of the directly added edge costs.
- The lifelong planning approach utilizes the concept of restarting searches from current node, so whenever there is an update to a node's intersection's t and g value, all nodes with them as predecessors get updated, causing a chain of changes within the values.
- The core of the algorithm lies in the speed function that calculates the speed for an intersection from it's t value. So instead of using the current congestion information for a an intersection, which may be occupied by the vehicle 1 hour later, using the estimate t value at that time makes sense, as it reduces computation to update the speeds dynamically. The only time there are any change in the predicted speeds and t values, are when there are unexpected road closures or accidents. We assume that our vehicle will reach our intersection at the time that is equivalent to the t value.

Algorithm 2: *LTA** Dependencies.

```
procedure CalculateKey(s)
    return [w*h(s) + min (g(s), rhs_g(s)),
    min (g(s), rhs_g(s))]
end procedure
procedure Initialize( )
    U=ϕ
    for all s ∈ S, $rhs\_g(s), g(s), rhs\_t(s) \leftarrow \infty$
    $rhs\_g(start)$ , $rhs\_t(start) \leftarrow 0$
    U.insert(start, [h(start), 0]
end procedure
procedure UpdateNode(v)
    if $(v \neq start)$
        $rhs\_t(v) = min_{s' \in pred(v)}(\frac{c(s',v)}{speed(v,s',T(s'))}) + T(s')$     ▷ T Value Updation
        $rhs\_g(v) = min_{s' \in pred(v)}(\frac{c(s',v)}{speed(v,s',T(s'))}) + g(s')$     ▷ G Value Updation
    end
    if $(u \in v)$
        U.remove(u)
    end
    if $(g(u) \neq rhs(u))$
        u.insert(u, calculateKey(u))
    end

end procedure
procedure ComputeShortestPath( )
    while (U.TopKey() < CalculateKey (S_goal)
    or rhs_g(S_goal) ≠ g(S_goal)
        or rhs_t(S_goal) ≠ t(S_goal))
                                            ▷ Termination Checking
        u = U.pop()
        if (g(u) > rhs(u))
            g(u) ← rhs_g(u)
            t(u) ← rhs_t(u)
            for all s ∈ succ(u), updateNode(s)
        end
        else
            g(u) ← ∞
            for all s ∈ succ(u) ∪ u, updateNode(s)
        end
    end
end procedure
```

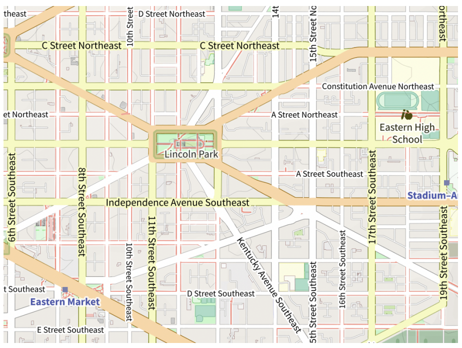

Figure 1: Road transport network of Washington City area.

3 RESULT AND ANALYSIS

3.1 Theoretical Analysis

3.1.1 Challenges

Incremental search algorithms like the one proposed are best suited for environment with changing features which in our case is the congestion present on the road. The algorithm proposed utilises the time derived values of speeds for different edges which are garnered through the t values. The algorithm currently runs on Restarting the A* search which requires the re-computation of edges for their g and t values whenever there is a change. Certain approaches from within the literature would seem logical to be applied in such cases to boost the search:

1. Bidirectional search: The following LTA* approach cannot be applied using a bidirectional A* because the core of the LTA* algorithm lies in using the time based speed of an edge to estimate a node's g_cost at a given time, and it is practically impossible to estimate the t_cost of node's in the opposite direction in the same order as they are from the start to finish.
2. D* Lite: Following the intuition from the bidirectional search as well, we can see that if the node encounters an unexpected blockage, which may occur in traffic systems, the recovery/repair part of the D* Lite from that time on can compute the shortest path from the goal to current position of the vehicle, as it designed to do, but it may encounter issues in computing the fastest path, due to failing to collect accurate time based speeds for each edge due to the t values getting changed due to changes in g values.

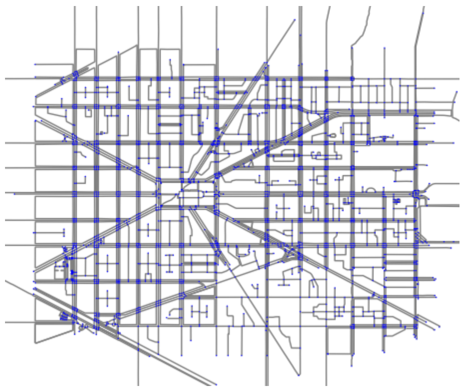

Figure 2: Graphical grid model of Washington City area.

3.2 Observation and Analysis

The proposed algorithm is implemented using a 500×500, 1000×1000, as shown in Table 1 Grid World systems with constant lengths equal to 200 meters, with $5-10\%$ nodes and corresponding edges randomly removed from the grid to mimic a real world like city road map. The algorithm is also tested on an urban area of Washington, US, depicted in Figure 1 and Figure 2 to validate the efficiency of the proposed methods. To calculate the distance between the nodes on the map, the longitudinal and latitudinal coordinates are used to compute the distance.

The analysis of time base values for the different times of the days has been conducted using random integer values for speeds generated between the range of 2m/s or 7.2km/hr, and 15m/s or 54km/hr and dynamically updated for an edge for 10 minute intervals within a day. The observations in all three examples taken (see Table 1) show that the shortest route every time isn't the fastest route, not that it isn't possible, which in contradiction with the popular belief and trend present in the literature where all algorithms focus on shortest routes. The numbers show how effective this algorithm can be in certain situations where shortest paths are extremely congested, and how choosing alternative paths as shown in examples can even reduce travel times by half.

Numerous observations were conducted, both on the grid world example as well as on the map of Washington, and it was found out that on an average, even though using a weight equal to w equal to 1.6 was used to overestimate the heuristic function, it rarely compromised with the optimal solution. The observation were run for various simulations and it was found that the number of nodes traversed decreased by almost a factor of 10 when employing a weight of 1.6. These results are recorded in the graphs, depicted in Figure 3, Figure 4, Figure 5, and Figure 6. This value

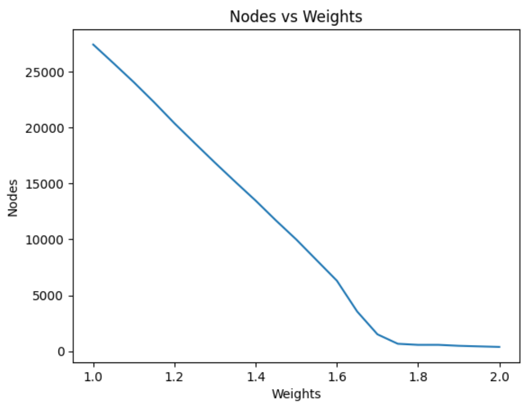

Figure 3: Nodes Searched in the Grid Vs Weight for LTA*.

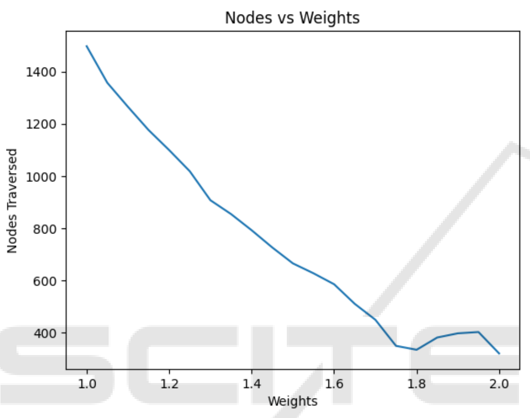

Figure 4: Nodes Searched on the Map Vs Weight for LTA*.

of the weight, might suit some grid-worlds or maps but isn't in any way a generalization to all kinds of maps.

4 CONCLUSION

This research addresses a critical aspect of urban transportation by focusing on the identification of the fastest least congested path in road transport networks using a novel timed weighted A^* algorithm. Through the modelling and implementation of LTA* that incorporates real-time traffic data and dynamic variables, this research work demonstrates a significant improvement in identifying optimal routes. The extensive simulations and case studies conducted on urban road networks provides valuable insights into the adaptability and robustness of the proposed algorithm to sudden road closures. The positive outcomes observed in 1000x1000 Grid Worlds demonstrates the algorithms potential to address the complex and dynamic nature of urban traffic, making it a practical tool for urban planners, transportation authorities, and

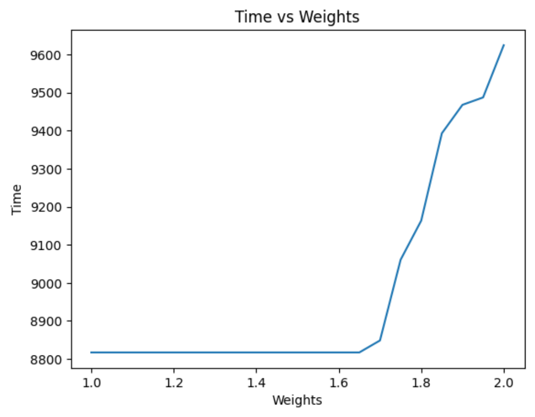

Figure 5: Fastest Time in the Grid Vs Weight for LTA*.

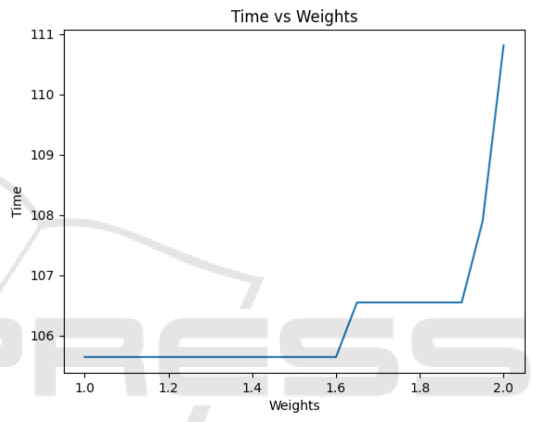

Figure 6: Fastest Time on the Map Vs Weight for LTA*.

Table 1: LTA* Observations on different test cases.

GridWorld: 500x500		
Observation	Distance	Time
A*	112.2Km	18563 s
LTA*	114.6Km	10484 s
A*	65.2Km	11186 s
LTA*	65.6Km	5789 s
GridWorld: 1000x1000		
Observation	Distance	Time
A*	143Km	23684 s
LTA*	144.6Km	13190 s
A*	222.6 Km	36723.9 s
LTA*	225.8 Km	20695.3 s
Washington Map		
Observation	Distance	Time
A*	1.788Km	275 s
LTA*	2.233Km	195 s
A*	1.554Km	221 s
LTA*	1.646Km	152 s

commuters seeking more efficient travel routes. This, in turn, aligns with the broader goal of achieving sustainable urban mobility and improving the overall quality of modern transportation systems.

The current LTA* algorithm works great in dynamic environments without actually restarting the search throughout the search space but for only the successor intersections where the edge speeds had been predicted wrongly. To further speed up the search we can employ algorithms like ANA* to produce even faster first outputs search results.

REFERENCES

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong planning a star. *Artificial Intelligence*, 155(1-2):93–146.

Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A., and Thrun, S. (2005). Anytime dynamic a*: An anytime, replanning algorithm. In *ICAPS*, volume 5, pages 262–271.

Ma, X., Tao, Z., Wang, Y., Yu, H., and Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197.

Pohl, I. (1970). Heuristic search viewed as path finding in a graph. *Artificial intelligence*, 1(3-4):193–204.

Richter, S., Thayer, J., and Ruml, W. (2010). The joy of forgetting: Faster anytime search via restarting. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 20, pages 137–144.

Stern, R., Felner, A., Van Den Berg, J., Puzis, R., Shah, R., and Goldberg, K. (2014). Potential-based bounded-cost search and anytime non-parametric a∗. *Artificial Intelligence*, 214:1–25.

Van Den Berg, J., Shah, R., Huang, A., and Goldberg, K. (2011). Anytime nonparametric a. In *Proceedings of the AAAI conference on artificial intelligence*, volume 25, pages 105–111.

Zhang, Q., Chang, J., Meng, G., Xiang, S., and Pan, C. (2020). Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1177–1185.