A Survey on Optimal Path Finding Algorithms for Pursuing a Moving Target

Renoy Zachariah[1], Rahul Jadhav[2], Rohit Mishra[3], Prem Kishan[4]

Computer Department, University of Pune/ Sinhgad Institute of Technology Lonavla

Abstract— **Path finding algorithms are widely used to find the shortest optimal path from a starting point to a destination point. The complexity of such an algorithm greatly depends on whether the target is in motion while the optimal path is being calculated or if the target is static. Finding the optimal path wherein the hunter and prey are stationary is easy, however pursuing a moving target is difficult as it poses challenges such as search space optimization, limited computational resources, partial knowledge about the environment and real time response. There has been a lot of research in this regard, each following a different approach towards finding the optimal path. In this paper we present an overview of the different approaches.**

***Keywords**—AI, Moving-target pursuit, path finding, A* search, optimisation*

I. INTRODUCTION

At its core, any path finding method starts by searching a graph by starting at one vertex and exploring it's adjacent nodes. This is repeated with the adjacent nodes until the destination node is reached. This path thus found is usually the shortest route to the destination. Two primary problems of pathfinding are to find a path between two nodes in a graph and to find the optimal shortest path. Basic algorithms such as Depth first and Breadth first search address the former problem by exhausting the possibilities; processing starts from the given node, iterating over all possible paths until the destination is reached.

In any path finding algorithm there is a hunter and a prey. These algorithms could be categorized into two parts based on whether the prey is stationary or in motion.However there are two main approaches using which we find the optimal path, they are static and dynamic. In a static algorithm information about the environment is known prior to any activity in the environment. This makes it possible to pre compute values necessary for finding the optimal path even before the hunter and prey are in motion. In a dynamic algorithm however, the environment may change while the computation is in process. An overview of techniques to find the optimal path in such situations is given by Leenen et al. [2] Pursuit tasks occur frequently in many domains. For instance, in computer games human-controlled agents are often pursued by hostile agents. In cooperative settings however, it is required that a computer controlled agent follow the other agents in the world.

Moving target pursuit (MTP) task arises in numerous domains, such as law enforcement, video games. MTP poses multiple challenges for several reasons. Firstly, in real-time scenarios, the time available for planning each move is limited, for example in video games, several game designers impose strict limits to maintain a fluid frame-rate. This frame-rate may differ from one designer to another but usually they're of the order of milliseconds for all path finding units. Secondly, even though a strategy such as 'go to the target's starting position and then repeat all of its moves' may guarantee a capture when the pursuers are faster than the target, we prefer an algorithm where the pursuer outsmarts the target instead of just following and outrunning it.

II. LITERATURE SURVEY

E. W. Dijkstra in his paper A Note of Two Problems in connexion with graphs [3] stated two problems wherein he considered n points (nodes), some or all pairs of which are connected by a branch; such that the length of each branch is already given. He stated a restriction wherein at least one path exists between any two nodes. The first problem dealt with construction of a tree of minimum total length between any n nodes. The second problem was to find the path of minimum total length between any two given nodes in the graph

Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute described an algorithm [4] as an extension to Edsger Dijkstra's 1959 algorithm [3]. This algorithm known as A* achieves better time performance by using heuristics. A* uses a BFS (best-first-search) and finds a least-cost path from a given initial node to a goal node (out of one or more possible goal nodes). A* traverses the graph by following a path of lowest expected total cost/distance, keeping a sorted priority queue of alternate path segments along the way. As stated above it uses a heuristic cost function of a node (f(x)) to determine the order in which the search visits each node in the tree. The cost function is a sum of two other functions:

(i) The past path cost function (g(x)): the known distance from the starting node to the current node x.
(ii) A future path cost function (h(x)): the admissible 'heuristic estimate' of the distance from x to the goal node.

Being an informed search algorithm it first searches the routes that appear to be most likely to lead towards the goal. A* also considers the distance already travelled into account

Scenarios such as route planning for a mobile robot require finding the lowest-cost path through a graph. While traversing if the arc costs change then this results in the replanning of the remainder of the path. A sensor-equipped mobile robot having incomplete information about its environment may work in this manner. As the sensors read additional data about the environment, the robot can revise its plan to reduce the total cost of the traversal. While the traversal path is being replanned, the robot could either move in another direction or wait for the correct path to be compute. The Dynamic a* (D*) algorithm finds optimal traversal paths in real-time by incrementally repairing path to the robot's state as new information is discovered. Anthony Stenz in his paper "The focussed D* algorithm for real-time replanning[5] proposed an extension to the D* algorithm in which the total time required for the initial path calculation and the subsequent replanning operations are significantly reduced. This extension to the original d* algorithm allow for full generalization of the A* for dynamic environments, wherein the arc costs change while traversing the solution path

The modelling of problems in terms of constraints has the advantage of a natural, declarative formulation. A CSP formulation of a problem states what must be satisfied, without specifying how it should be satisfied. It consists of a set of variables, and a set of constraints. Each constraint is defined over a subset of the set of variables. Louise Leenen and Alex Terlunen in their paper [1] used a CSP approach to solve the Military Unit Path Finding Problem (MUPFP). In which the problem is to find a path from a starting point to a destination where a military unit has to move or be moved, safely whilst avoiding obstacles and threats and minimising the path cost in a digital representation of the actual terrain. Leenen and Alex proposed an algorithm that uses a heuristic to focus the search for an optimal path. Since existing approaches used to solve the path planning problems focussed on combining the path costs with various other criteria such as obstacle avoidance. The authors approach is to optimize only the path costs while ensuring that other criteria such as safety requirements are met by adding constraints. The author's approach is based on the algorithm originally proposed by Stenz [6] . The CSP approach is to optimise the path costs while ensuring that certain other criteria such as safety requirements are met. The objective function used is a pure cost function. A constraint based approach is adopted with a clear distinction between the goal of obtaining an optimal path cost and satisfying safety measures. Constraint programming approach is followed as it allows flexibility in terms of modelling different constraints. If a new requirement has to be met, it can be done by adding a suitable constraint or modifying the existing constraint to model the new problem

S Koenig and M Likhachev in their paper [7] described a way in which the search operation could be performed in real-time since characters in real time computer games needed to move smoothly. The paper described a way of speeding up the repeated A* searches having the same goal states, it did this by updating the heuristics between A* searches. This technique is then used to develop a novel real time heuristic based search method called as Real Time Adaptive A*, this algorithm can choose local searches in a fine-grained way. It quickly updates the values of all states in its local search space. Experiments showed that following this technique allows RTAA* to follow smaller cost trajectories for given time limits per search episode

Pursuing a moving target poses as one of the most challenging problems occurring frequently all the while consuming computational resources ineffectively. The artificial intelligence engine in these environments is responsible for delating hostile agents to plan their pursuit paths over large game maps, this is in order to chase the other escaping weak agents despite being bounded by constraints such as real time interactivity, sense scope, moving speed, avoiding obstacles and target adversarial escaping strategy. Dave C Pottinger in his paper [8] showed that in small scale games such as Age of Empires 2 (Ensemble Studios 1999) 61% - 70% of the simulation time was spent in path finding, most of which was utilized for pursuing moving targets.

Sven Koenig in his article [9] categorized the real-time search or local search into an agent centered search. He stipulated that agent-centered search methods involve interleaving the planning and execution phases and restrict planning to the part of the domain around the current state of the agent. An example could be the current location of a mobile robot or the current board position of a game. He stated that such methods could execute actions in the presence of time constraints and are often associated with a small sum of execution and planning cost, this is because they allow agents to gather information early in the non-deterministic domain, this reduces the amount of planning required for unencountered situations. He demonstrated the design and properties of several agent-centered search methods, focussing on robot exploration and localization.

Real time heuristic search poses as a challenging variant of agent-centered search as the agent's planning time per action is bounded by a constraint independent of the size of the problem [10]. Path finding in modern computer games pose such restrictions as a large number of units must plan their paths simultaneously over a map. Search algorithms such as A*, IDA*, D*, ARA*, AD* cannot be considered as real time and they may lose completeness when a constant bound is imposed on per action planning time. Real time search algorithms on the other hand retain completeness but tend to produce unacceptable suboptimal solutions. Dynamic control in real time heuristic search [ ] extends the real time search algorithms with an automated mechanism for dynamic depth and subgoal selection. These new algorithms find path within 7% of optimal while on a average expanding roughly a single state per action.

S Koenig, M Likhachev and X Sun, in their paper [11] study moving target search where an agent (hunter) has to catch a moving target (prey). This is in a scenario in which the agent is unaware about the terrain initially but can observe it within a certain sensor range around itself.

The strategy used by them made the agent move on a shortest presumed unblocked path towards the target. They studied how the agent could find the shortest path faster by performing repeated A* searches. They extend the Adaptive A* search method [7] to moving target search and demonstrate that the resulting Moving Target (MT) Adaptive A* is faster than isolated A* searches. They demonstrated in their experiments that MTAA* is faster than D* Lite by one order of magnitude for moving target search in known and unknown maps/mazes. This is if both the search methods use the same informed heuristics.

X Sun, S Koenig and W Yeoh in their paper [12] proposed an algorithm Generalized Fringe Retrieving A* which uses the incremental search algorithm Fringe Retrieving A* to solve moving target search problems on arbitrary graphs. They use an incremental search method in which the hunter follows a cost minimal path from its current state to the target's current state and replans a new path whenever the target moves off the path. This process is repeated until the hunter is in the same state as the target at which point the target is caught. FRA* is an incremental version of A*, it repeatedly finds shortest path for moving targets in a known grid world. It does this by repeatedly transforming the previous search tree to the current search tree [13]. The algorithm starts A* with OPEN and CLOSED lists obtained from previous searches. It uses geometric properties specific to two-dimensional grids, due to which they cannot be used for arbitrary graphs. The authors generalize it to a Generalized FRA* (G-FRA*) which helps solve moving target search problems on arbitrary graphs.

In their paper Moving Target Pursuit Algorithm Using Improved Tracking Strategy [1] the authors propose a novel tracking algorithm called Target Automatic Optimization Moving Target Pursuit (TAO-MTP) to effectively address the challenges in problems consisting of single hunter and single prey. In that TAO-MTP uses a queue to store the prey's trajectory and at the same time runs Real Time Adaptive A* (RTAA*) [7] to reach the optimal position which is updated periodically within limited steps in the trajectory. This helps in minimizing the overall pursuit cost. The algorithm works by making the hunter move to any position in the explored trajectory (this position may or may not be the optimal position) and then moves along the trajectory to catch the prey. The authors stated that as long as the hunter's moving speed is greater than that of the prey and it's sense scope is large, it will eventually catch the prey

CONCLUSION

This paper is a survey on different path finding, moving target pursuit algorithms and methodologies that were proposed by researchers for better development in the field of Artificial Intelligence – Path finding. Each researcher makes use of a previously stated algorithm and improves on it. In future we propose to use the approaches specified in [1] [2] [7] to devise a novel moving target pursuit algorithm that finds the optimal shortest path every time and handles obstacles in real time.

REFERENCES

[1] Mingliang Xu, Zhigeng Pan, Hongxing Lu, Yangdong Ye, Pei Lv, and Abdennour El Rhalibi. "Moving Target Pursuit Algorithm Using Improved Tracking Strategy." *IEEE Transactions on Computational Intelligence and AI in Games, VOL. 2, NO. 1, MARCH 2010*

[2] L. Leenen, A. Terlunen, and W. le Roux, "A Constraint Programming Solutionfor the Military Unit Path Finding Problem", ser. Mobile IntelligentAutonomous Systems: Recent Advances. Bota Raton, USA: Taylor &Francis Group, 2012.

[3] E. W. Dijkstra, "A note on two problems in connexion with graphs,"*NumerischeMathematik*, vol. 1, pp. 269–271, 1959

[4] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristicdetermination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*,vol. 4, no. 2, pp. 100–107, Jul. 1968

[5] A. Stenz, "The focused D* algorithm for real-time replanning," in *Proc.Int. Joint Conf. Artif. Intell.*, 1995, pp. 652–1659

[6] A. Stentz, "Optimal and efficient path planning for partially-knownenvironments," in *Proceedings of the IEEE International Conference onRobotics and Automation (ICRA)*, vol. 4, 1994, pp. 3310–3317

[7] S. Koenig, M. Likhachev "Real-Time Adaptive A*" in *Proc. Int.Joint Conf. Autonom. Agents Multiagent Syst.*, 2006, pp. 281–288

[8] D. C. Pottinger, "Terrain analysis in realtime strategy games," in *Proc.Comput. Game Develop. Conf.*, 2000, pp. 763–782

[9] S. Koenig, "Agent-centered search," *AI Mag.*, vol. 22, no. 4, pp.109–132, 2001

[10] V. Bulitko, M. Lˇustrek, J. Schaeffer, Y. Bjornsson, and S. Sigmundarson,"Dynamic control in real-time heuristic search," *J. Artif.Intell. Res.*, vol. 32, pp. 419–452, 2008

[11] S. Koenig, M. Likhachev, and X. Sun, "Speeding up moving-targetsearch," in *Proc. 6th Int. Joint Conf. Autonom. Agents Multiagent Syst.*,Honolulu, HI, 2007, article no. 188

[12] S.Koenig, X. Sun, and Y.William, "Generalized adaptive A*," in *Proc.7th Int. Conf. Autonom. Agents Multiagent Syst.*, 2008, pp. 469–476

[13] X. Sun, W. Yeoh, and S. Koenig, "Efficient incremental search formoving target search," in *Proc. Int. Joint Conf. Artif. Intell.*, 2009, pp.615–620