Incremental Heuristic Search Algorithm in Path Planning of Autonomous Robot

Christzen Leonardy 13517125
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517125@std.stei.itb.ac.id

***Abstract* — Robotics is a branch of engineering that involves the conception, design, manufacture, and operation of robots. Robotics is an interdisciplinary branch of engineering and science that includes mechanical engineering, electronic engineering, information engineering, computer science, and many others.**

One of the branches of robotics which involve path planning (or navigation problem) is autonomous-robot engineering. One must assess the skill of understanding how the robot move, how it must perceive its environment in the ideal way so it can move efficiently, both in time to process its environment, and move with minimum cost.

This paper will give an overview on application of incremental heuristic algorithm, a combination of both incremental and heuristic search algorithm, focusing on Lifelong Planning A* in autonomous-robot path planning. This is important in domains that are changing dynamically.

Keywords — incremental, heuristics, path planning, autonomous, robot

I. INTRODUCTION

It is often important that searches be fast. Some have developed several ways of speeding up searches by trading off the search time and the cost of the resulting path. In this article, we discuss a different way of cutting the cost of searches, namely incremental search. Incremental search is a search technique for continual planning (or, synonymously, replanning, plan reuse, and lifelong planning) that reuses information from previous searches to find solutions to a series of similar search problems potentially faster than is possible by solving each search problem from scratch. Different from other ways of speeding up searches, it can guarantee to find shortest paths.

Most search algorithms replan from scratch, meaning, it solves the new search problem independently of the old ones. However, this can be inefficient in large domains with frequent changes and thus limit the responsiveness of the analyses that they can perform. Fortunately, the changes to the search problems are usually small. A robot, for example, might have to replan when it detects a previously unknown obstacle, a traffic routing system might have to replan when it learns about a new traffic jam, and a decision-support system for marine oil-spill containment might have to replan when the wind direction changes. This suggests that a complete recomputation of the best plan for the new search problem is unnecessary since some of the previous search results can be reused. This is what incremental search does.

Incremental search solves dynamic shortest path problems, where shortest paths must be found repeatedly as the topology of a graph or its edge costs change. The idea of incremental search has been around for some time. The idea of incremental search has also been pursued problems other than path finding. For example, a variety of algorithms have been developed for solving constraint logic programming problems where the constraints change over time. In this article, however, we give overview of incremental search only in the context of dynamic shortest path problems.

To increase the popularity of incremental search in robotics path planning, some requirements must be met. First, one needs to devise more powerful incremental search algorithms than those that currently exist. Second, one needs to study their properties more extensively, both analytically and experimentally, to understand their strengths and limitations better. Third, one needs to demonstrate that they apply to some applications and compare them to other search algorithms for these applications to demonstrate that they indeed have advantages over them. This article describes one particular incremental search algorithm and its application, and then discusses its potential advantages and limitations.

II. THEORIES

A. Robot

A robot is a machine — especially one programmable by a computer — capable of carrying out a complex series of actions automatically. Robots can be guided by an external control device or the control may be embedded within. Robots may be constructed on the lines of human form, but most robots are machines designed to perform a task with no regard to how they look.

Robots can be autonomous or semi-autonomous and range from humanoids such as Honda's Advanced Step in Innovative Mobility (ASIMO) and TOSY's TOSY Ping Pong Playing Robot (TOPIO) to industrial robots, medical operating robots, patient assist robots, dog therapy robots, collectively programmed swarm robots, UAV drones such as General Atomics MQ-1 Predator, and even microscopic nano robots. By mimicking a lifelike appearance or automating movements, a robot may convey a sense of intelligence or thought of its own. Autonomous things are expected to proliferate in the coming decade, with home robotics and the autonomous car as some of the main drivers.

B. Autonomous Robot

An autonomous robot, or automatic robot, is a robot that performs behaviors or tasks with a high degree of autonomy. Autonomous robotics is usually considered to be a subfield of artificial intelligence, robotics, and information engineering. Early versions were proposed and demonstrated by author/inventor David L. Heiserman.

Autonomous robots are particularly desirable in fields such as spaceflight, household maintenance (such as cleaning), waste water treatment, and delivering goods and services.

Some modern factory robots are "autonomous" within the strict confines of their direct environment. It may not be that every degree of freedom exists in their surrounding environment, but the factory robot's workplace is challenging and can often contain chaotic, unpredicted variables. The exact orientation and position of the next object of work and (in the more advanced factories) even the type of object and the required task must be determined. This can vary unpredictably (at least from the robot's point of view).

One important area of robotics research is to enable the robot to cope with its environment whether this be on land, underwater, in the air, underground, or in space. A fully autonomous robot can:

- Gain information about the environment
- Work for an extended period without human intervention
- Move either all or part of itself throughout its operating environment without human assistance
- Avoid situations that are harmful to people, property, or itself unless those are part of its design specifications

An autonomous robot may also learn or gain new knowledge like adjusting for new methods of accomplishing its tasks or adapting to changing surroundings.

C. Path Planning

Motion planning or p ath planning (also known as the navigation problem or the piano mover's problem) is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement.

For example, consider navigating a mobile robot inside a building to a distant waypoint. It should execute this task while avoiding walls and not falling down stairs. A motion planning algorithm would take a description of these tasks as input, and produce the speed and turning commands sent to the robot's wheels. Motion planning algorithms might address robots with a larger number of joints (e.g., industrial manipulators), more complex tasks (e.g. manipulation of objects), different constraints (e.g., a car that can only drive forward), and uncertainty (e.g. imperfect models of the environment or robot).

A motion/path planner is said to be complete if the planner in finite time either produces a solution or correctly reports that there is none. Most complete algorithms are geometry-based.

D. Incremental Search Algorithm

In computing, incremental search, incremental find or real-time suggestions is a user interface interaction method to progressively search for and filter through text. As the user types text, one or more possible matches for the text are found and immediately presented to the user. This immediate feedback often allows the user to stop short of typing the entire word or phrase they were looking for. The user may also choose a closely related option from the presented list.

The method of incremental search is sometimes distinguished from user interfaces that employ a modal window, such as a dialog box, to enter searches. For some applications, a separate user interface mode may be used instead of a dialog box.

E. Heuristic Search Algorithm

In computer science, artificial intelligence, and mathematical optimization, a heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution. This is achieved by trading optimality, completeness, accuracy, or precision for speed. In a way, it can be considered a shortcut.

A heuristic function, also called simply a heuristic, is a function that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow. For example, it may approximate the exact solution.

F. Incremental Heuristic Search Algorithm

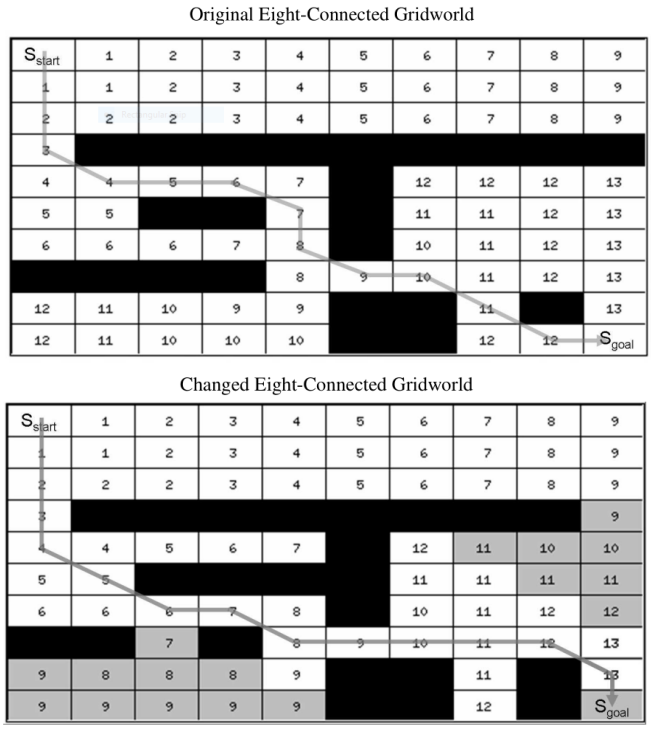

Figure 1 Simple Gridworld

One way of speeding up searches is incremental search. A different way of speeding up searches is heuristic search. The question arises whether incremental and heuristic search can be combined. Many of the start distances that have changed in the example from Figure 1 are irrelevant for finding a shortest path from the start cell to the goal cell and thus do not need to get recalculated. Examples are the cells in the lower left corner of the gridworld. Thus, there is a potential advantage to using heuristics to avoid having to recalculate irrelevant start distances. To summarize, there are two different ways of decreasing the search effort of breadth-first search for finding the start distances for the changed gridworld from Figure 1:

- Incremental search algorithms, such as DynamicSWSF-FP (Ramalingam and Reps 1996a), find shortest paths for series of similar search problems potentially faster than is possible by solving each search problem from scratch (complete search) because they do not recompute those start distances that have not changed.
- Heuristic search algorithms, such as A* (Nilsson 1971), use heuristic knowledge in the form of approximations of the goal distances to focus the search and fi nd shortest paths for search problems faster than uninformed search because they donot compute thosestartdistances thatare irrelevant for finding a shortest path from the start cell to the goal cell.

Consequently, we developed a search algorithm that combines incremental and heuristic search, namely Lifelong Planning A* (LPA*) (Koenig and Likhachev 2002b). We call it "lifelong planning" in analogy to "lifelong learning" (Thrun 1998) because it reuses information from previous searches. LPA* repeatedly finds shortest paths from a given start vertex to a given goal vertex on arbitrary known finite graphs (not just gridworlds) whose edge costs increase or decrease over time (which can also be used to model edges or vertices that are added or deleted). The pseudo code of the simplest version of LPA* reduces to a version of A* that breaks ties among vertices with the same f-value in favor of smaller g-values when used to search from scratch and to DynamicSWSF-FP when used with uninformed heuristics. In fact, it differs from DynamicSWSF-FP only in the calculation of the priorities for the vertices in the priority queue (Line {01} in the pseudo code of Figure 3). It is unoptimized and needs consistent heuristics (Pearl 1985). We have also developed more sophisticated versions of LPA* that are optimized (for example, recalculate the various values much more efficiently than the simple version), can work with inadmissible heuristics, and break ties among vertices with the same f-value in favor of larger g-values. These changes make LPA* more complex.

procedure CalculateKey(s)
{01} return $[\min(g(s), rhs(s)) + h(s); \min(g(s), rhs(s))]$;

procedure Initialize()
{02} $U = \emptyset$;
{03} for all $s \in S$ $rhs(s) = g(s) = \infty$;
{04} $rhs(s_{start}) = 0$;
{05} U.Insert$(s_{start}, [h(s_{start}); 0])$;

procedure UpdateVertex(u)
{06} if $(u \neq s_{start})$ $rhs(u) = \min_{s' \in pred(u)}(g(s') + c(s', u))$;
{07} if $(u \in U)$ U.Remove(u);
{08} if $(g(u) \neq rhs(u))$ U.Insert$(u,$ CalculateKey$(u))$;

procedure ComputeShortestPath()
{09} while (U.TopKey()$<$CalculateKey(s_{goal}) OR $rhs(s_{goal}) \neq g(s_{goal})$)
{10} $u =$ U.Pop();
{11} if $(g(u) > rhs(u))$
{12} $g(u) = rhs(u)$;
{13} for all $s \in succ(u)$ UpdateVertex(s);
{14} else
{15} $g(u) = \infty$;
{16} for all $s \in succ(u) \cup \{u\}$ UpdateVertex(s);

procedure Main()
{17} Initialize();
{18} forever
{19} ComputeShortestPath();
{20} Wait for changes in edge costs;
{21} for all directed edges (u, v) with changed edge costs
{22} Update the edge cost $c(u, v)$;
{23} UpdateVertex(v);

Figure 2: Lifelong Planning A* Pseudocode

III. CONCLUSION

Incremental heuristic Search algorithm is a fast replanning method for robot navigation in unknown terrain that combine incremental search and heuristic search. The algorithm search from the goal vertex towards the current vertex of the robot, use heuristics to focus the search, and use a way to minimize having to reorder the priority queue.

Incremental heuristic Search in the heuristic part builds on our LPA*, a strong similarity to A*, is efficient (since it does not expand any vertices whose values were already equal to their respective goal distances) and has been extended in a number of ways.

Incremental heuristic search can provide auto-robot an efficient way to plan its path and reach its goal, by determining, dynamically as it makes its move, the shortest path available, and replans incrementally when the path is not traversable because of obstacles.

IV. Acknowledgment

A special note of thanks to Dr. Masayu Leylia Khodra ST, MT, my most inspiring lecturer at Bandung Institute of Technology for this interesting assignment that has broaden my knowledge on appliance of incremental heuristic search algorithm in robotics. I would also like to thank my friends and the internet communities for the readings and other sources of my inspirations that helped me get this paper done.

References

[1] Sven Koeing, Maxim Likhachev, Yaxin Liu, David Furcy. 2004. "Incremental Heuristic Search in Artificial Intelligence". College of Computing - Georgia Institute of Technology

[2] Heiserman, David. 1979. "How to Build Your Own Self-Programming Robot". TAB Books.

[3] G. Ramalingam and Thomas Reos. 1992. "An Incremental Algorithm for a Generalization of the Shortest-Path Problem". Computer Science Department-University of Wisconsin. Madison.

[4] J. Jaffar and J. L. Lassez. 1987. "Constraint Logic Programming". ACM. New York .

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 April 2019

Christzen Leonardy 13517125